MIT/LCS/TR-233

AUTOMATIC EXTENSION OF AN AUGMENTED TRANSITION NETWORK GRAMMAR FOR MORSE CODE CONVERSATIONS

Gail E. Kaiser

This research was supported by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under [redacted]

This blank page was inserted to preserve pagination.

Automatic Extension of an Augmented Transition Network Grammar for Morse Code Conversations

Gail E. Kaiser
March 1980

This research was supported by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract number N00014-75-C-0661.

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts 02139

This empty page was substituted for a blank page in the original document.

Table of Contents

This empty page was substituted for a blank page in the original document.

Abstract

This report describes a 'learning program' that acquires much of the knowledge required by a parsing system that processes conversations in a 'natural' language akin to ham-radio jargon. The learning program derives information from example sentences taken from transcripts of actual conversations, and uses this knowledge to extend the 'core' augmented transition network (ATN) grammar. The parser can use the extended grammar to process the example sentences, plus a large number of syntactically and semantically related sentences.

The learning program uses a set of heuristics to determine the difference between the existing version of the grammar and a superset that could process the example sentence. A set of models act as templates to produce possible extensions to the grammar. An evaluation measure selects one of the extensions and adds it to the grammar. This extension is henceforth an integral component of the knowledge base and may be used by the parser to process conversations and by the learning program to extend the grammar further.

This report relates the mechanisms used by the learning program to grammatical inference of context-sensitive languages, which include the natural languages, and some proposed linguistic models of human language acquisition. These models describe language acquisition as a process of developing hypotheses according to the constraints of innate universal rules, and acceptance of those hypotheses that make it possible for the child to understand new sentences. Similarly, the learning program develops its hypotheses within the constraints of certain 'universal' models and accepts only those hypotheses that enable the parser to process the motivating example.

Acknowledgements

I would like to thank the following people for their contributions to this report: Al Vezza, for recognizing the necessity of a mechanism to extend CATNIP's knowledge base as new transcripts become available, and for his advice and support while I pursued this idea; Tim Anderson and Stu Galley, for reading various versions of this report; Dave Lebling, for help muddling through the MDL environment; Dave Sherry, for developing the original CATNIP; Janet Schoof, for helping me finally get this report in print; and David Dill, for many useful suggestions for the implementations of CATNIP and MAGE, and for the acronym 'MAGE'.

This report is an expansion of a thesis of the same name that was submitted to the Department of Electrical Engineering and Computer Science on 11 May 1979, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering. The thesis supervisor was Albert Vezza, Research Associate in the Laboratory for Computer Science. This research was supported by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract number N00014-75-C-0661.

The author's current address is: Gail E. Kaiser, Carnegie-Mellon University, Computer Science Department, Schenley Park, Pittsburgh, PA 15213.

keywords and phrases: augmented transition networks, language acquisition, Morse code

List of Illustrations

1. Introduction

1.1 Motivation

As computer technology advances, computers are being applied to more complex tasks that require increasingly greater 'domain-specific' knowledge. One of the pressing goals of computer science and engineering is to determine how to incorporate this knowledge into computer systems in an efficient way.

There are two major approaches in current use that attempt to solve this problem. One approach in current use is the development of various 'tools' specifically tailored for installing the domain-specific knowledge, including very-high-level languages and special-purpose editors. Another approach, which has met with considerably less success, is to let the computer do most of the work of acquiring the information. This report describes a computer program that acquires much of the knowledge necessary to perform its task.

The task in this case is parsing human conversations in a very limited domain. The conversations take place between operators on Morse code radio-networks in a simple 'natural' language akin to ham-radio jargon, where the possible topics of conversation are limited by radio network protocol to such things as establishing contact, discussing and sending messages, re-sending garbled parts of the messages, and ending contact. In tandem with a transcription system, the parser processes the hand-sent Morse code to produce a human-readable transcript and information summary. The domain-specific knowledge required by the parser consists of the discourse structure and the syntax and semantics of the language, and this knowledge is organized as an augmented transition network (ATN).

However, the programmer who developed the original parser was not able to incorporate enough domain-specific knowledge into the system to parse all, or even most, of the actual conversations that occur in this domain, simply because this information is not available in its totality. However, one can expect that as the parser performs its task, transcripts of conversations that it can not process adequately with its current knowledge base will become available. It was desirable to develop a mechanism by which the system could extend its knowledge base, given the new transcripts, in a way that enables it to correctly process each of the new transmissions (or sentences) in these example conversations, plus a large number of similar transmisslons.

A computer program with these abilities would incorporate a high degree of learning ability. Winston [23] describes the levels of learning ability as a shift of effort from the teacher to the student. His four levels include learning by being programmed, learning by being told, learning by example, and learning by discovery. The original domain-specific knowledge incorporated by the programmer into the system described in this report is an example of 'learning by being programmed'. A system that was explicitly guided by some teacher in its acquisition of knowledge, with the instructions of the teacher phrased in the *language of the domain* rather than some programming language, would be 'learning by being told'. The program described here at times must 'learn by being told', for the program must sometimes ask questions of the human supplying examples and the human responds in the language of the domain. However, for the most part this program 'learns by example': the program derives the ability to parse new sentences and phrases from the examples of sentences and phrases presented to it.

One approach to developing a computer program that could acquire such knowledge, or 'learn', from examples is to borrow from theories about the learning processes of humans, the most successful 'learning machines' to date. However, the human learning processes are incompletely understood. Current theories suggest that they consist in part of forming generalizations from data and deriving rules from them. The correct application of these rules by the learner demonstrates that something has, indeed, been learned.

One well-known example of human learning that seems, on the surface, very similar to the problem at hand is the acquisition of language by children. Humans acquire their first language almost entirely by hearing it spoken. The generalization of data follows very quickly as children learn to produce grammatical sentences with no formal instruction in the grammar of their native language; they infer the rules of their grammar from the sentences they hear spoken [13].

Some linguistic models proposed by Chomsky [4, 5] make the controversial proposition that a child may *know* about certain aspects of language: some knowledge is innate and the child need not learn these aspects in the usual sense. These innate aspects of language are called the *universal grammar* and, according to these models, form the basis for forming generalizations and deriving rules from the utterances that the child hears spoken.

The system described here borrows some aspects of these linguistic models that seem particularly appropriate for extension of the grammar used by the parsing program, and incorporates them in a separate learning program that includes all the domain information of the original parser and can operate on the same grammar. This does not mean that the resulting computer program models human language

acquisition in any psychologically realistic sense. However, the research described here demonstrates that theories that attempt to explain human learning processes are also useful for developing computer programs that acquire knowledge.

Previous work in this area has concentrated on the development of algorithms for the inference of formal grammars from very large sets of examples. The problem of inferring an exact grammar for an arbitrary (but constrained) language has been solved for the regular languages [3, 12, 14], and some very restricted subsets of the context-free languages [6, 7, 8, 17, 22]. However, there has been very little progress toward the development of a general and practical mechanism for deriving grammars for the more powerful context-sensitive languages, which include all natural languages. This research represents a step toward this goal.

1.2 Organization

The result of this research is a learning program called MAGE (Morse Automatic Grammar Extension system). MAGE uses a 'domain model' that includes information about the simple language and the environment in which it is used, a small 'core' grammar organized as an ATN, and some knowledge about what type of result it is expected to produce. MAGE is designed to receive individual examples of sentences from the language and extend the grammar so that it can parse each example, plus a large number of similar sentences. An arbitrary number of examples may be provided to produce an arbitrarily large grammar.

MAGE uses a set of heuristics to determine the difference between the grammar and a superset of the grammar that would be able to process the example sentence. It uses a set of models as templates to 'enumerate', or list, a set of possible extensions to the grammar that might bridge this difference. A unique 'evaluation

measure' guides the enumeration process, to keep the list of possible extensions workably short, and selects one of these extensions, which is then added to the grammar. The evaluation measure is based on the ability of the grammar to extract important information from conversations: an extension is enumerated only if it provides a mechanism for parsing the new phrase, without considering the context, and an extension is selected only if it makes it possible for the entire example containing the new phrase to be parsed by the standard ATN parsing algorithm that is used as a tester.

The process outlined above is analogous, in some aspects, to linguistic models developed by Chomsky [4, 5] and Dale [9] of the learning mechanisms used by children when acquiring a native language. According to these models: the child has innate knowledge of a universal grammar that provides a mold in which the child develops the grammar for her own language; and the child uses a set of universal rules that prescribe the ways she can organize the utterances she hears and evaluate the hypotheses she forms according to whether or not they help her to *understand* the utterance. These components of the language acquisition models are similar to the domain model, hypothesis-formation models, and evaluation measure of MAGE, respectively.

Although MAGE borrows from linguistic models, this author does not necessarily endorse any of these models nor support these or any other linguistic theories. The augmented transition network mechanism discussed in this report is not related to these linguistic models, nor does this author claim that the ATN is a realistic model of human language comprehension. What this report does say about these theories of language acquisition is that some aspects of the models can be

implemented as a computer program operating on a data structure representing an ATN grammar.

The rest of this report is organized as follows:

- Chapter 2 presents MAGE's domain model and the particular aspects that make possible the evaluation measure.
- Chapter 3 states the general grammatical inference problem, and presents the hypothesis-formation algorithm and evaluation measure used by MAGE in its partial solution to the related problem of grammatical extension.
- Chapter 4 discusses further the domain model, hypothesis-formation models, and evaluation measure in the context of language acquisition by children.
- Chapter 5 describes the design and implementation of MAGE.
- Chapter 6 contains a summary and conclusions.

2. An ATN with Semantic Categories

2.1 Machine Recognition of Hand-sent Morse Code

The research was motivated by the real-world problem of automating the recognition and understanding of hand-sent Morse code in an amateur-radio network environment. Morse code consists of five elements: dots, dashes, mark spaces, letter spaces, and word spaces. The English alphabet, digits, and punctuation are encoded as groups of one to six marks (dots or dashes) separated by mark spaces. These groups are separated from each other by letter spaces (ideally, three times as long as a mark space) and combined into words, which are separated from each other by word spaces (ideally seven times as long as a mark space). For example, "SOS" is transmitted as "dot ms dot ms dot ls dash ms dash ms dash ls dot ms dot ms dot ws", where "S" is encoded as "...", "O" as "---", "ms" means mark space, "ls" letter space, and "ws" word space. Morse code is transmitted over radio by short signals (dots) and long signals (dashes), with the pauses in between signals serving as spaces.

It is desirable to automate the reception of these signals and the transcription of the marks and spaces back into character text, to produce a readable output. However, there are many aspects of manual Morse code that make transcription difficult, not only for a machine but also for a human operator. Many errors are introduced by radio attributes like transmitter chirp and atmospheric interference, and by sender irregularities including spacing errors (e.g. a letter space that is shorter than a nearby mark space), mark errors (e.g. sending a dash instead of two dots) and spelling errors. The result is analogous to speech that is slurred or broken

by arbitrary pauses and includes a few mispronounced words.

Research in machine transcription of manual Morse code began in the 1950's and included the development of MAUDE (Morse AUtomatic DEcoder) [11] at M.I.T.'s Lincoln Laboratory. MAUDE and other early transcribers were based on a small set of statistical and linguistic rules; no attempt was made to take advantage of the constraints provided by radio network protocol or the informational content of the transmissions.

Recently, a system called COMCO-1 (COmputerized Morse Code Operator) [21] has been developed at M.I.T.'s Laboratory for Computer Science. It involves a new perspective on the manual Morse code problem: it utilizes extensive knowledge of the peculiarities of hand-sent Morse code and amateur-radio network protocol, and attempts to 'understand' the Morse code conversation.

COMCO-1 consists of three components: a signal-processing system, a Morse-code-to-character-text transcriber, and a text understander, or parser. The signal-processing system produces a file of mark and space durations based on its analysis of radio signals.

The transcriber, a software system called COMDEC (COmputerized Morse DECoder), converts marks and spaces to character text using a set of modules, each of which is an 'expert' on one aspect of transcription. Each module corrects certain types of errors and makes additions to a set of suggested transcriptions, where each transcription consists of a list of vocabulary elements. COMDEC is aided by dictionaries of ham-radio jargon and the English language.

2.2 An ATN Parser for Morse Code Conversations

The parser, called CATNIP (Comco-1 Augmented Transition Network Interfaced Parser) [16], uses an augmented transition network (ATN) grammar to evaluate the transcriptions suggested by COMDEC with respect to their syntactic and semantic coherence and selects one that matches a path through the ATN. The grammar includes a transition network that represents the syntactic/semantic structure of a Morse code conversation, and a set of registers, and functions that operate on them, designed to store information extracted from a conversation. Both COMDEC and CATNIP are written mostly in MDL [15], a high-level programming language of the LISP family.

The conversations largely consist of a shorthand language called *chatter*. Network protocol and the limited vocabulary of chatter constrain the possible topics of conversation to the statement and query of operator identification, signal characteristics, rendezvous information, message traffic information, and so forth. The conversations are task oriented, and a parser can 'comprehend' the dialogue because both the topic of conversation and the movement from topic to topic is severely limited.[1] No formal definition or language generator exists for this natural-language-like jargon, so the grammar was derived from several hours of transcripts.

This grammar follows the ATN formalism described by Woods [24]. An augmented transition network consists of two components: a transition network (TN), and a set of registers with associated functions. A transition network is a set of

[1] An example of a short but typical conversation that can be parsed by CATNIP is given in Appendix I.

'named' finite state machines, or *subnetworks*, where a transition symbol may be the name of another (or the same) subnetwork. When the name of some subnetwork appears as one of the symbols of a transition, it indicates a 'push' to that subnetwork, in the sense of calling a subroutine. A terminal state indicates a 'pop' to the 'calling' transition, which may then be followed to the state it designates. When other words appear as transition symbols, the parser operates the subnetwork as a finite state machine, attempting to 'accept' the input sequence.

An ATN also includes a set of *registers* designed to hold contextual information, a set of *tests* that determine the validity of a word in a given context, and a set of *actions* to change the contents of the registers as the context shifts. A possibly empty set of tests and actions is associated with each transition. When a transition symbol has been matched by one of the mechanisms described above, the transition may be followed only if each of the tests can be passed.

After the parser has been determined that a transition may be followed, each of the associated actions is applied before the parser continues processing from the next state. Actions are often used to build and connect parts of parse trees, which are saved in the registers until completed at the end of the parse, but this ability is not used by CATNIP. Augmented with registers, tests, and actions, a transition network has the power of a Turing machine. A more detailed discussion of augmented transition networks is given by Ritchie [19].

CATNIP's grammar conforms very closely to Woods' definition of an ATN, with two exceptions. The first is that CATNIP's registers, and the tests and actions that act on them, were designed to manipulate the particular informational items that are expected to appear in chatter conversations, rather than to build parse trees for

legal sentences. These items include call-signs (names) and locations of operators; time and date; ratings of strength, clarity, etc. of signals; traffic information like message number, length of message, and the message body; and conversation history like pending questions and requests.

This exception illustrates one of the most powerful features of the augmented transition network model: the possibility exists of adding to the model whatever facility is needed and seems natural to do the job. An addition requires only a relaxation of the restrictions on the types of tests and actions but no reformulation of the basic model.

2.3 The Syntax of Chatter

The second exception to the standard ATN is the unusual organization of CATNIP's grammar into topical categories. Each of the nineteen subnetworks is designed to process a particular set of *semantically* related substrings. ATN knowledge bases for language processing are usually organized into subnetworks that process *syntactic* structures, such as 'noun phrase' and 'verb phrase' in English. A subnetwork begins processing a substring when it is referenced by a 'push specification' (i.e. the name of the subnetwork) on a transition of a higher-level subnetwork. The push specification performs the dual role of expressing a top-down prediction that some particular kind of item is needed at that point in the input stream, and indicating which subnetwork is to be used to process the item. The suitability of a particular type of category (for example, 'noun phrase' is a syntactic category) depends both on the ways that grammatical predictions can be phrased and on the classes of items that can be processed in a similar fashion (i.e. by the same subnetwork).

It has been suggested by Ritchie [19] that this 'subroutine' mechanism presupposes a syntactic organization of the grammar into subnetworks and that a semantic organization could not be viable, "since semantic categories are not the appropriate organizational units for an augmented transition network grammar." However, I have found that the addition of meaning-based categories is not only justified, but also superior to using only syntactic categories for embedded structure processing in the Morse code radio network domain.

The chatter language is sufficiently limited, little syntax exists, and what does exist is either weak or can be described in more revealing terms as a result of semantic considerations. The language consists of only four generic types of words: *q-signs*, *pro-signs*, *call-signs*, and *abbreviations* [2]. Q-signs are internationally agreed-on abbreviations which were devised for radiotelegraph use. Each q-sign represents a complete thought; e.g. "QSK" means "I can hear you between my signals; break in on my transmission" and "QTQ ?" means "Can you communicate with my station by means of the International Code of Signals?" The first letter in every q-sign is 'Q'. Pro-signs, or procedure signals, also have precise definitions but do not express complete thoughts and are closely related to network protocol; for example, "AS" means "wait" or "stand by", and "AR" means "end of transmission". Call-signs are station identifiers and serve as names of radio operators. The final category consists largely of simple abbreviations of commonly used English words and phrases; for example: "GA" means "go ahead", "NR" means "number", "OK" means "okay" and "PSE" means "please". The frequency of these English abbreviations is so low that an English-like syntax model could not be developed for chatter.

There are two types of syntactic rules. The first is characterized by the following example: if either of the constructs "callsign DE callsign" ("Station *<call-sign1>*, this is station *<call-sign2>*") or "DE callsign" ("This is station *<call-sign>*")[2] occurs in a transmission, it occurs near the beginning of that transmission. A 'transmission' is equivalent to a 'sentence' in spoken conversation, and it does not necessarily include everything transmitted by a single operator between signals from other operators.

The second type of syntactic rule is the order of the 'arguments' that follow almost all q-signs and many other words, e.g. "QSL MSG NR 3 ?" ("Can you acknowledge receipt of message number three?") and "QRZ ROCK 3500" ("You are being called by Rock on frequency 3.500 kHz"). The definition of each q-sign includes a set of informational 'slots' that should be filled by the q-sign's arguments (for example, "QRZ" alone means "You are being called by - - - on frequency - - - kHz"). However, "QRZ 3500 ROCK" is just as meaningful as "QRZ ROCK 3500", and the phrase may be transmitted both ways, so order isn't really very important here. It is clear from these examples that these syntactic rules can easily be reformulated in terms of the underlying semantics. The only syntax rule that seems very strong is the fact that arguments always follow the word of which they are arguments.

The first two constructs discussed above are different ways of identifying a new operator as she begins transmission. Either can occur in any position where self-identification of an operator is desired: logically this is at the beginning of a

[2]CATNIP's grammar uses the convention that any word in lower-case letters is a generic token, which is replaced by an appropriate chatter word at parse-time.

transmission by that operator. The syntactic rule is replaced by a more intuitive semantic rule that groups the two phrases in the topical category "Identification of Operators", denoted **ID-OP** in the grammar.

The number, type, and ordering of the argument words not only depend on the lexical features of the particular word of which they are arguments but also are a function of the context. For example, in the phrase "NR 1 GR 200 QTR 1500" ("[message] number 1, with 200 groups, at 1500 hours"), "GR" is followed by the number of English words or code-groups in the next message. However, in a transmission like "PSE RPT GR 10 , 20 , 30 OK ? K" ("Please repeat code-groups 10, 20, and 30. Okay? Over"), the arguments of "GR" are one or more numbers separated by delimiters, referring to the previously sent code-groups in positions *<number1>*, *<number2>*, ..., *<numberN>*. Thus the syntax of a word's arguments depends on the current topic of discussion.

The potential of syntactic rules is further weakened by the spoken-language aspects of chatter conversations, for example, the existence of *noise* words. These include chatter words from both the pro-sign and abbreviation categories -- such as "R" ("roger"), a pro-sign, and "NW" ("now"), an abbreviation -- that an operator often sends as 'filler' while she is deciding what to say next. So another syntactic rule might be that a noise word can appear anywhere in a transmission, except as the last word in that transmission. However, most potential noise words can also appear as meaningful words in various contexts, for example "R" might be the response to "QRO ?" ("Shall I increase transmitter power?").

Noise words can appear at any time, because they are meaningless; this is a semantic rather than syntactic consideration, so this rule may be reformulated as a

semantic rule that allows meaningless words to appear in any context and requires them to be disregarded by the information-accumulating mechanisms of the parser.

2.4 The Semantic Structure of Chatter Conversations

Although the syntax of chatter is weak, there is a strong semantic structure imposed on Morse code conversations by radio network protocol. First, the operators involved must establish contact with each other, and this is represented by the **CONTACT** subnetwork in the ATN. Next, one operator prepares to send some message, and then sends it, either as code-groups or English text; this is represented by the **TRAFFIC** subnetwork.

Immediately following the sending of traffic, the receiver may ask to have several words repeated and eventually acknowledges receipt of the message. This process is modeled by the **REQ-INFO** subnetwork. The **TRAFFIC** and **REQ-INFO** subnetworks are repeated until all operators have sent all their prepared messages. Then the operators begin signing off, which usually involves negotiations regarding re-establishment of contact at some future time: this is represented by the **END-CNCT** subnetwork. At this point, the conversation may terminate, or one of the operators may continue by trying to establish contact with a new operator.

With one major exception, these four topics are the only possibilities for discussion and they always occur in this rigid order. The exception is the 'Interrupt Subnetwork', denoted **INTRUPT** in CATNIP's grammar, which can be pushed to (called) from <u>any</u> state and represents an interruption in the smooth flow of transmission. The possible types of interruptions include a third operator suddenly breaking in on a conversation; sudden static on the air waves, which must be dealt with by changes in transmitter characteristics or frequency; and so on. These

interruptions are very difficult to parse since the context is made invalid by the break, and this presents an interesting problem for the parser designer. However, to make the problem addressed in this report more tractable, I have ignored the 'interruption problem'.

The four main areas of discourse are broken down into additional subnetworks based on topical categories. For example, **CONTACT** has transitions indicating pushing to (calling) the lower-level subnetworks **ID-OP** (identification of operators), **NET-RELAY** (relay of operator identification through the network controller), and **QUAL-CNCT** (discussion of signal characteristics). It is only within these lowest-level subnetworks that syntactic structure shows up, for example, in the ordering of q-sign arguments, but, as discussed above, this structure results from semantic as well as syntactic considerations.

The semantic category of a push (call) specification fulfills its role as a top-down prediction that a particular topic will be discussed at that point in the conversation, and of course it indicates which subnetwork is to be used to process phrases discussing that topic. Semantic categories are more suitable for this application than syntactic categories due to the limited syntax of chatter and the strong protocol constraints on the discourse structure of a conversation.

The semantic organization of this ATN grammar not only is very unusual but also plays a unique role in the partial and limited solution to the grammatical inference problem discussed in the next chapter.

3. Grammatical Inference of ATNs

3.1 The Grammatical-Inference Problem

Scientists have been using formal linguistics for modeling natural and programming languages for over twenty years [14]. Grammars have been employed to describe the syntax of languages like chatter and can be used to characterize a syntactic source that generates all the sentences in a language. It would be useful if the grammar could be directly inferred from a set of sample sentences in the language in question. The process of deriving a grammar from a set of examples is called *grammatical inference*.

The general grammatical-inference problem is simply stated. Assume the existence of a source that generates strings of the form $x = a_1a_2...a_n$, where x is a sentence in a language L and each a_i is a word in the lexicon of L. L is assumed to possess some unique structural features that can be modeled by a grammar G. The grammatical-inference machine is given a finite set S^+ of sentences that are in L, and possibly another finite set S^- of sentences that are not in L. Using this information, the machine must infer the syntactic rules of the unknown grammar G.

The first difficulty encountered is the necessity of obtaining extra information in order to find an appropriate set S^-. Although the set S^+ can be obtained from the source, the set S^- can be defined only if an external teacher, who knows something about the properties of G, is available. Unfortunately, without S^-, the grammatical-inference problem is unsolvable except for a small number of highly constrained grammars [8]. The chatter language has this problem, because, with no formal definition, there is also no algorithmic means for determining that a given

string of chatter words is not likely to be transmitted over Morse code networks, or even for deciding whether a given word (that is not a q-sign) is in the chatter vocabulary.

Even though it is impossible for a grammatical-inference machine to find exactly one grammar for most languages without this negative information, it is often possible to enumerate a large set of possible grammars and then narrow down the solution in some way to a single grammar. A grammar is 'possible' in this sense if it accepts the sample. The problem of narrowing down the state-space to one grammar has been solved for regular languages, the very simple languages that can be generated by regular expressions and accepted by finite state machines (FSMs).

The limited case of regular languages is solvable because two finite state machine grammars that generate the same language are *equivalent*. Since all of the accurately enumerated grammars are equivalent, only one need be constructed, and it is the correct solution. Feldman et al. discuss the concepts involved [12], and two algorithms are presented by Biermann and Feldman [3].

However, these algorithms cannot be utilized to extend the grammar for chatter, since the nesting features of natural language are not adequately represented by finite state machine grammars. Chatter can be considered a natural language, because its representation requires nested structures, which are represented by the subnetworks of the ATN knowledge base,[3] and because it is an evolving, 'spoken' language.[4]

---

[3] Section 2.4

[4] The similarities between chatter and natural languages like English are discussed further in Section 4.3.

The context-free languages are more powerful than regular languages, because they can model the self-embedding and nesting properties of natural and programming languages. They can be represented by grammars whose production rules are of the form **A** --> *a*, where **A** is a single nonterminal symbol and *a* is a string of terminal symbols and nonterminal symbols [1]. A *terminal symbol* is an element from the language being modeled. Since the left-hand side of the rule contains a single symbol, no context is necessary to determine the derivation of a sentence. Context-free languages are accepted by transition networks (TNs).

It is considerably more difficult to derive grammars for context-free languages than for regular languages, because an infinite number of possible grammars can be enumerated for any set of data. No algorithm exists that can decide whether two arbitrary context-free grammars accept the same language. Some mechanism is needed that limits the number of grammars produced to a tractable level and then selects one of them that is 'best'. Such a mechanism is termed an 'evaluation measure'.

One approach to solving this problem is to look for a reasonably good fit, with some suitable definition of 'reasonable', rather than trying to find a grammar that generates exactly the input sample. Cook states [7] that an infinite language, i.e. any language that includes an infinite number of sentences, assures a discrepancy between a grammar inferred from a finite sample and the grammar for the language. He used a cost function measuring the tradeoff between decrease in complexity and increase in discrepancy to bound his machine's search-space. The machine described by Wharton [22] uses a similar evaluation measure, but it receives its examples via a multi-step method rather than all at once; this methodology tends to

increase the efficiency of enumeration but cannot guarantee minimum complexity in the ultimate result.

Another approach is to require a human 'teacher' to guide the grammatical inference machine as it enumerates possible grammars and select the 'best' grammar according to some subjective measure. In the scenario developed by Knobe and Knobe [17], the teacher is a knowledgeable person who provides individual examples in optimal order with optimal variety, and who can recognize grammatical and ungrammatical strings without knowing the formal grammar for the language. The machine enumerates first general and then more specific productions, and each production is tested by the teacher as it is enumerated. The machine retains the most general rule that does not produce any strings ruled illegal by the teacher. This scenario places a heavy burden on the teacher to present an adequate 'course'.

A third approach, described by Crespi-Reghizzi [8], attaches structural descriptions to the examples. This limits the number of hypotheses that are compatible with the data and thus reduces the enumeration problem. The extra information, although similar to the type of information required by the complexity/discrepancy measure and the teacher's judgements above, must be justified, since it departs from the standard model of grammar acquisition. Crespi-Reghizzi explains that this structural information is similar to the stress and intonational information available to a child acquiring a natural language, and that the widespread belief that there must be a partially semantic basis for the acquisition of syntax implies the availability of some structural information to the learner of a language. Of course, the availability of structure vastly reduces the number of

alternative possible grammars and assures that the acquired grammar generates sentences with structures consistent with their meaning.

The grammatical-inference machines described above are all successful for subsets of the context-free languages. However, there is as yet no algorithm that can infer the complete set of rewriting rules from a positive sample of an arbitrary context-free language [6]. It is not surprising that no general mechanism has been developed for grammatical inference of supersets of the context-free languages, particularly the context-sensitive languages, which include all natural languages.

Now, the context-sensitive are even more powerful than the context-free languages. They can be represented by grammars with production rules of the form *a* --> *b*, where both *a* and *b* consist of any number of terminal and nonterminal symbols; the length of *a* must be less than or equal to the length of *b* [1]. Since the left-hand side of a rule may include more than one symbol, context is necessary to determine the derivation of a sentence. The context-sensitive languages are accepted by augmented transition networks (ATNs). All natural languages are members of the set of context-sensitive languages: contextual information is necessary to parse constructs such as reflexives and relative clauses in English.

3.2 Grammatical Inference and MAGE

This report describes a grammatical-extension machine for an augmented transition network grammar for a very limited 'natural' language. Since augmented transition networks represent and are equivalent to the context-sensitive grammar, the development of MAGE is a small step toward a general solution to the very difficult problem of inference for context-sensitive grammars.

There are three ways in which this machine's model of the

grammatical-inference problem diverges from the standard model discussed in the first section of this chapter. The first is that MAGE's grammar is not inferred from scratch but builds on a core grammar, which includes a small transition network, a set of pre-coded functions for the tests and actions, and a dictionary of q-signs (but not other chatter words).

The second difference is that the grammar is extended incrementally; that is, each example is successfully learned before the next example is provided. This makes the inference problem more difficult than usual, because MAGE cannot exploit structural similarities between examples when determining the embedded structure of the grammar. The incremental feature is necessary in the Morse code domain, because a structurally complete sample is required in order to derive a complete grammar for any language [6]; a positive sample of a language is *structurally complete* if each rewriting rule of the grammar is used at least once in the generation of the sample. It is impossible to generate a structurally complete sample of chatter, because no formal grammar exists, and the language is continuously evolving. In other words, since the grammar can never be complete, the extension mechanism must always be ready to add one more example to the grammar.

The third difference is a result of the second: the extension procedure is not expected to result in an *exact* grammar for the language that is equivalent to some known formal definition. The best that the system can do, given the constraints of the domain, is to generate an extended grammar that understands all sentences it received as examples, plus a large number of similar sentences.

Keeping in mind these deviations from the standard grammatical-inference

model, the computer program the author has developed is successful at what it tries to do: extend an augmented transition network grammar for the limited Morse-code domain. MAGE is an *enumerative* procedure in the sense that it considers many potential additions to the grammar for each example it is supplied. However, the evaluation measure guides the enumeration of possible extensions, and each enumerated extension is selected or rejected before the next extension is postulated. As soon as one extension has been approved, the enumeration process halts. Thus all but the ultimate result are rejected before any data structure is generated. Since only one 'physical' grammar exists at any point in time, and extensions result in physical alterations of this data structure, the program may be considered a *constructive* model.

3.3 Hypothesis Formation and Selection

There are two phases to the hypothesis-formation/hypothesis-evaluation process. The first is the selection of a structural extension to the transition network, to result in a grammar that can *accept* the current example. The second is the specification of a set of tests and actions to be attached to the transition network, to enable the parser to *understand* the current example. These processes are independent and sequential, and they are presented here separately.

MAGE operates on a transition network grammar (it ignores the tests and actions during this phase) consisting of thirteen topically categorized subnetworks.[5] Given an example transmission, or an example conversation containing one or more

[5]The 'Interrupt Subnetwork' and the five related subnetworks of CATNIP's grammar are not part of MAGE's core grammar, because the current version of MAGE does not deal with the interruption problem.

speaker changes,[6] the program first determines if the example is already accepted by the grammar, by attempting to parse it. MAGE tries to match the example to the grammar using a standard transition network parsing algorithm, with one deviation: rather than requiring a single start-state, the parser performs a depth-first search from several potential start-states, including all states that can precede the beginning of a transmission. An example should not begin in mid-transmission, although the program can handle this in some instances. The parse is *nondeterministic*, i.e., conceptually it follows many paths in parallel (although it actually uses a depth-first search), because the grammar may contain more than one subnetwork representing the same subsequence of tokens or words, as do many TN and ATN grammars.

If the example is already accepted by the grammar, the program prints an appropriate message and asks for another example. If the first word or words of the example are accepted by one or more subnetworks, but the following word does not match any transition leaving the last state of any of these partial paths, the hypothesis-formation procedure takes control with pointers to the 'last-matched-states' and the next word in the example. The same sequence of words may be accepted by more than one subnetwork, because the parse has multiple start-states and the grammar is inherently nondeterministic. If the first word of an example is not accepted by any transition leaving any start-state, the set of 'last-matched-states' in this case consists of the possible start-states discussed above, and the next word in the example is the first one.

[6] A 'speaker change' occurs in a Morse-code conversation when one operator ceases transmitting Morse code and another begins.

At some point in the example, marking the end of the new phrase, the words of the example resume matching the symbols on the transitions of the TN. This may happen at more than one state, for the reasons stated above. If the new phrase is at the end of the example, it matches any terminal state in the transition network by default. The state(s) containing the transition(s) where the path resumes and the terminal state(s) matched by default are called the 'end-of-phrase' states. The task now is to add some structural representation of the words between these matches (the new phrase) to the transition network component of the grammar. MAGE uses the models presented below to accomplish this objective.

The set of models represents all single-transition extensions to the general three-state finite state machine shown in Figure 1, with several exceptions: it is undesirable to return to a start-state from some other state in the subnetwork except in a small number of prescribed circumstances; it is preferable for a subnetwork to contain a terminal state, and then repeat the entire subnetwork, rather than return from that state to the start-state. A single subnetwork without tests and actions is an FSM. Model 0 (Figure 1) represents the original status of a subnetwork: the circle containing *S* is a start-state; the circle with the darkened area is a terminal state; the single intermediate state represents the arbitrarily complex web of states and transitions between the start-state and a terminal state in an actual subnetwork.

Each of the models illustrated in Figures 2 through 8 represents a general one-transition extension to model 0. All extensions that are possible, considering the chatter domain, are included in this set. Any of the three circles in these models that correspond to the original circles in Model 0 may represent a 'last-matched-state' and any terminal state may represent the 'end-of-phrase', depending on the

particular model and circumstances; a circle other than the original three always represents a new circle added to the subnetwork as part of the extension. Since each example is expected to include only one new phrase, only one type of extension is actually used for each example. However, the new phrase generally consists of more than the single word that can be attached to a single transition. The transition can be viewed as modeling a string of transition/next-state pairs, with the first transition in this string leaving a state in the original subnetwork as shown in the model, and the final transition connected to the next-state shown in the particular model.

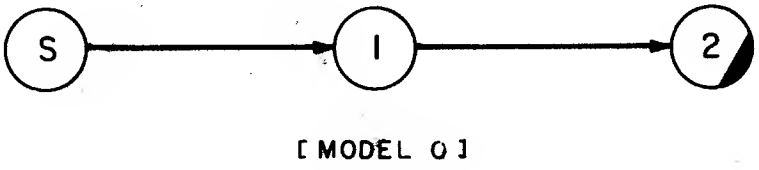


Figure 1: Model 0
General subnetwork

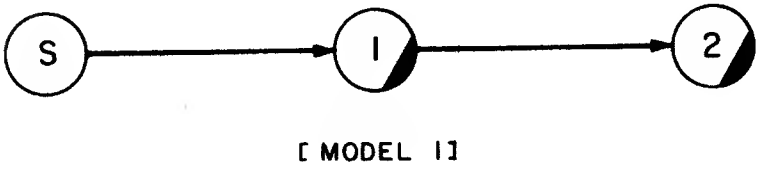
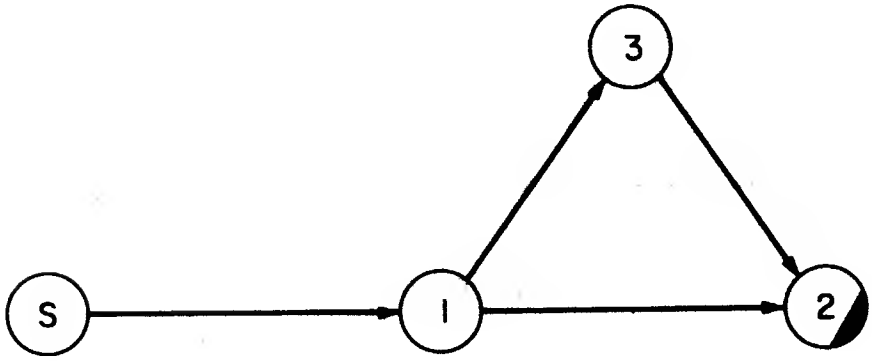


Figure 2: Model 1
'Last-matched-state' becomes terminal state

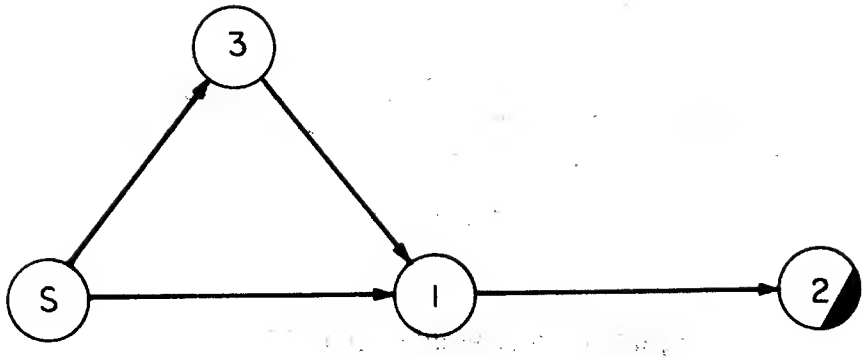
[MODEL 2]

Figure 3: Model 2
A terminal state that is also a 'last-matched-state' becomes a possible intermediate state



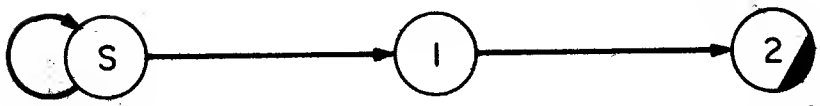
[MODEL 3]

Figure 4a: Model 3
State(s) inserted parallel to transition between adjacent states, which are 'last-matched-state' and 'end-of-phrase'



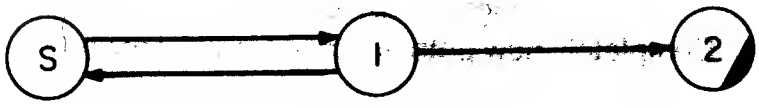
[MODEL 3]

Figure 4b: A special case of Model 3



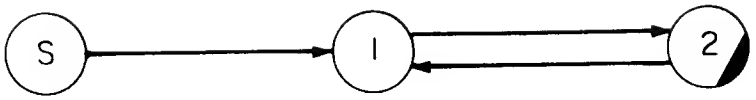
[MODEL 4]

Figure 5: Model 4
Transition loops to same state, which is both
'last-matched-state' and 'end-of-phrase'



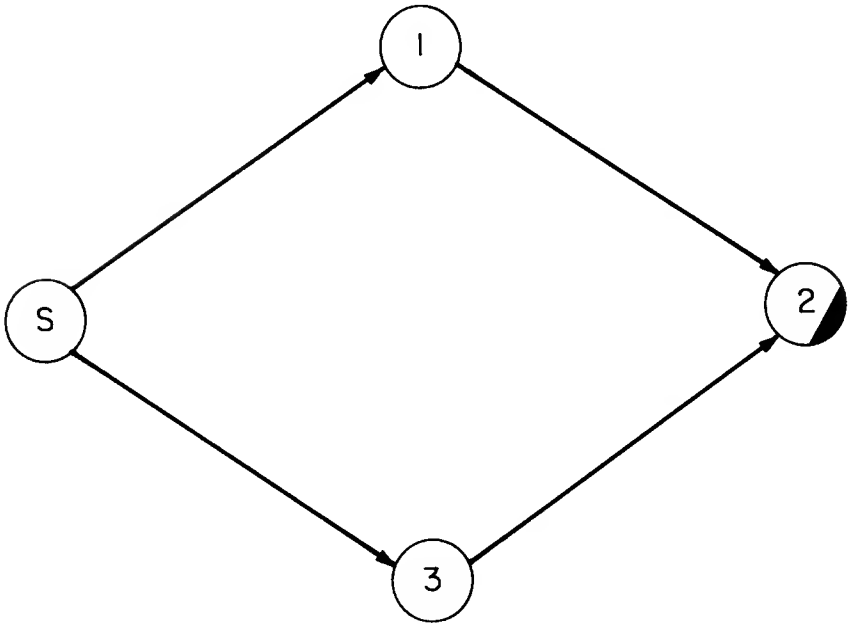
[MODEL 5]

Figure 6a: Model 5
Transition returns from 'last-matched-state'
to previously visited state



[MODEL 5]

Figure 6b: A special case of Model 5



[MODEL 6]

Figure 7a: Model 6
Completely new path is formed in subnetwork

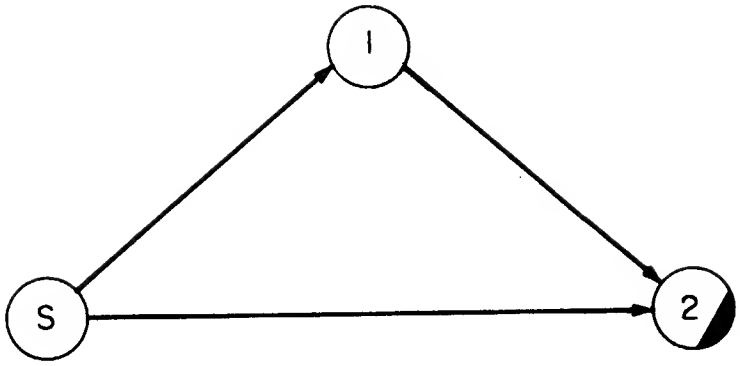


Figure 7b: A special case of Model 6

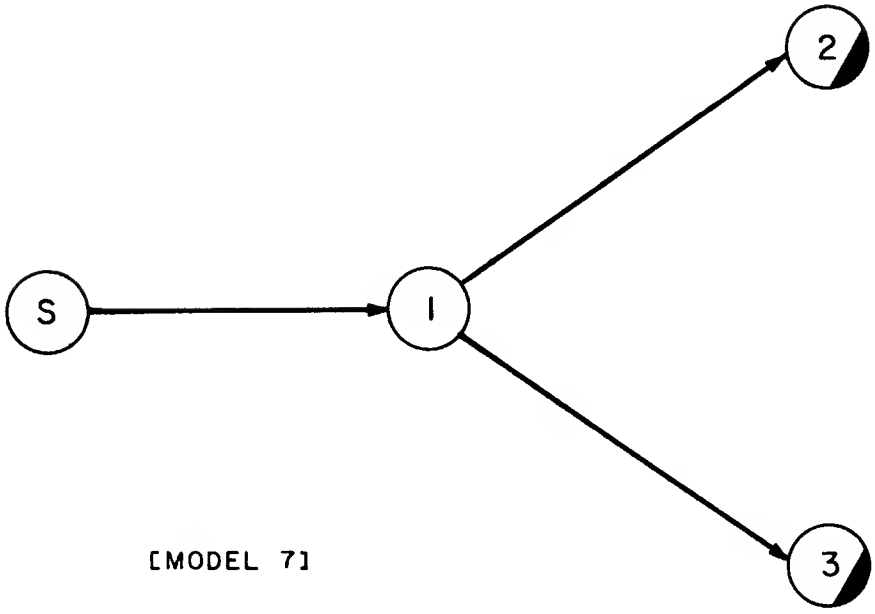


Figure 8: Model 7
New transition added between 'last-matched-state'
and new terminal state

MAGE compares each model to each last-matched-state/end-of-phrase pair.

The hypothesis-formation procedure enumerates a set of model/pair combinations called 'templates', matching particular states in the model to the last-matched-state and end-of-example of the pair. The first component of the evaluation measure guides this process, restricting it to enumerating only those models that provide a means for accepting the new phrase in the finite state machine sense: the first word in the phrase matches some symbol attached to a transition leaving the start-state of the extension derived from the model, the second word matches some transition leaving the state pointed to by the transition for the first word, and so on. The state pointed to by the transition matching the last word in the new phrase must either be a terminal state or contain a transition that matches the first word in the rest of the example, which follows the new phrase.

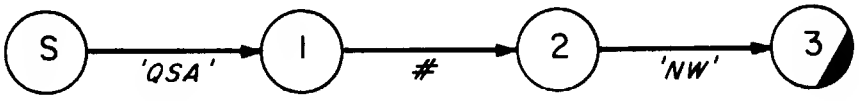
If there is only a single last-matched-state, and only one of the above models provides a mechanism for accepting the new phrase, then this model is subjected immediately to the second component of the evaluation procedure. If this model also provides a mechanism for accepting the new phrase in the context of the current example, i.e., the entire example would be accepted by the core grammar plus this extension, then the evaluation is said to 'succeed', and the extension is physically added to the data structure representing the transition network component of the grammar. In this case, the test/action phase of the hypothesis-formation mechanism begins operation. If the evaluation fails, the example is rejected as unlearnable.[7]

If the structure of the example matches one of the above models, but there are

[7]The author has not found any actual transmissions that contain phrases that cause MAGE to fail, with the exception of transmissions containing one of the interruptions discussed in Section 2.4.

several last-matched-states, then the evaluation measure selects the first of these states that passes its criteria. This selection is justified, because in nearly all instances the first passing state is the only one: conflicts are prevented by a strict ordering of the start-states via the subnetwork in which each appears.

There are several situations in which more than one model is represented in the templates produced by the hypothesis-formation process, and in these cases the evaluation measure must select a model as well as a particular state pair. Consider the example "QSA 5 NW QTC K" ("The strength of your signals is excellent now. I have messages for you") diagrammed below, and assume that "QSA 5 QTC K" ("The strength of your signals is excellent. I have messages for you") is already accepted by the grammar. It is not clear during the hypothesis-formation stage whether to apply model 2 (Figure 9a) or model 3 (Figure 9b). Therefore, both of these possibilities are passed to the evaluation measure, which chooses between them on the basis of which transmission 'makes sense'.



[EXAMPLE]

Figure 9a: Model 2 applied to example

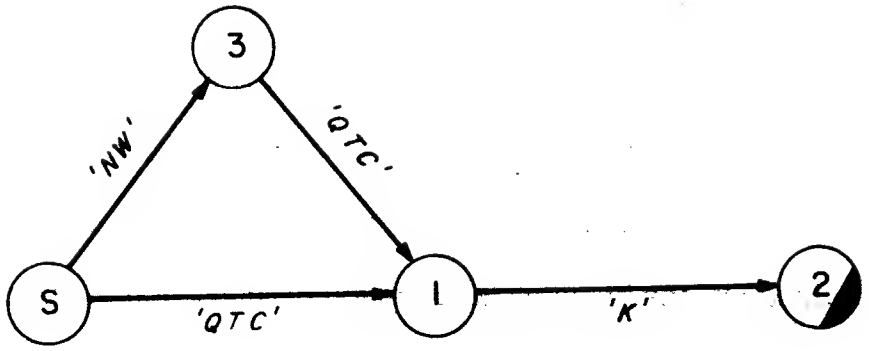


Figure 9b: Model 3 applied to example

In this case, model 2 wins, because the evaluation measure decides that "NW" refers to "QSA" rather than to "QTC". "QSA 5 NW" ("The strength of your signals is excellent now") is a plausible update to an earlier transmission like "QSA 1" ("The strength of your signals is very poor").[8] However, "NW QTC" ("Now I have messages for you") would not make sense unless the operator had previously transmitted something like "Wait. I will have messages for you soon": this statement cannot be made with the phrases contained in the core grammar. Of course, the extended grammar still accepts "QSA 5 QTC K" because the terminal state following the generic token " # " is not deleted. In fact, nothing is ever deleted from the core grammar; the only alterations performed by MAGE are additions.

After a specific structural hypothesis has been selected by the evaluation measure, the machine enters its second hypothesis-formation phase and

---

[8] If "QSA 5" is accepted by the core grammar, "QSA 1" is also, since the generic token '#' matches any number.

enumerates a set of potential test and action specifications for each transition of the new extension. If the symbol on a new transition is a q-sign, those actions associated with q-signs are enumerated; none of the prepared tests should be associated with transitions whose symbols are q-signs. The q-sign actions put information conveyed by q-signs and their arguments in certain registers:

⟨quality-of-contact⟩
⟨pending-questions⟩
⟨expected-actions⟩
⟨general-situation-description⟩.

If the transition symbol is some other type of word, but not a 'noise' word, the entire set of non-q-sign actions is enumerated. These actions put information in other registers, including:

⟨information-about-receiving-operator⟩
⟨information-about-sending-operator⟩
⟨id-number-of-message⟩
⟨number-of-words-in-message⟩
⟨number-of-words-received-so-far-in-message⟩
and others described in Appendix III.

If the symbol is a generic token, i.e. "callsign", "any", " # ", "delim", or "location", the entire set of tests is passed to the evaluation measure. These tests serve as filters to ensure that the chatter word that matches a generic token is reasonable in context, to prevent every random word from matching "any", for example, since this symbol is intended to match only code-groups or English words in a message body. The tests and actions to be associated with the new transition(s) are now selected by the evaluation measure.

3.4 A Unique Evaluation Measure

The evaluation component of MAGE is rather unusual in that it does not incorporate a cost function or other complexity-related consideration to select the 'best' from among the set of possible structural extensions or test/action specifications, nor does it use some subjective measure produced by a human teacher. Instead, the evaluation measure is based on the semantic organization of the augmented transition network grammar.[9]

The criterion for selecting a structural extension is simply: "Will this structural extension place the new phrase in the correct topical context?". Similarly, the criterion for augmenting a transition with a particular test or set of tests is: "Will this test or set of tests ensure that all words accepted by this transition are meaningful in the current context?". An action or set of actions is approved for a transition if those actions will select and save the important information contained in the phrase and ignore any meaningless words.

The first criterion is fairly simple to implement for phrases containing at least one q-sign, because all q-signs are associated *a priori* with appropriate topics represented by subnetworks. There are usually two or three subnetworks in which a particular q-sign might make sense, but the context of the rest of the example provides enough information to uniquely determine the topical category of the phrase.

Those phrases that contain neither q-signs, nor other words that are known to be synonymous with a particular q-sign (e.g. "RPT" ("repeat") is synonymous with

---

[9] Section 2.4

"QSM" ("Please repeat – – –")), are more difficult to evaluate. When an example contains an unknown word, MAGE asks the user if it is a synonym of any known word and, if so, which one.[10] Either the new word has a known synonym, or one or more of the other words in the phrase have known meanings that can be used to determine the meaning and topic of unknown words; this topical relation is used to place the new phrase in the appropriate subnetwork (i.e. context).

The selection of tests and actions proceeds along similar lines. Most tests are attached only to transitions with a generic symbol; most actions are attached to transitions with the symbol "new-speaker" (denoting a speaker change), a generic symbol, or a q-sign. In addition, the evaluation measure may attach actions to most symbols in the REQ-RPT subnetwork (request for something to be repeated and response to request) even though they were not generated during the hypothesis-formation phase; it is desirable to store any repeat request until it has been answered, regardless of how the request was phrased. This is one of the many semantic considerations dealt with by the evaluation measure.

The likely number and type of arguments for each q-sign are part of the machine's domain model, and they can be looked up in a table. This knowledge is used to attach actions to the transitions of q-sign arguments that convey information that should be stored in some register. In some cases, however, a q-sign may appear with a totally unexpected set of arguments, and the appropriate actions must be inferred from knowledge about the arguments themselves. The generic tokens "call-sign" and "any" appear in only a small number of contexts (the ID-OP and

---

[10] User-machine interaction is discussed further in Section 5.2.1.

NET-RELAY subnetworks and as q-sign arguments, and the MESSAG and REQ-RPT subnetworks, respectively), so their tests and actions can be effectively pre-programmed.

The major difficulty is with the generic token " # ", which can appear in almost any context and almost always has some important meaning. Fortunately, " # " is often preceded by some other word with associated test/action information that can be transferred to its argument. But in many cases there is no way of obtaining this information except to compare the particular use of " # " with its appearance elsewhere in the grammar, and to borrow the actions associated with the closest fit. This method is actually very successful at selecting the same set of actions that I would have selected by hand.

After this component of the evaluation measure has approved a set of test/action specifications for each transition in the previously selected structural extension, the specifications are attached to the extension and the data structure representing the ATN grammar is 'permanently' altered to include the completed extension. The addition is permanent in the sense that it can now aid in a future bootstrap process as described above.

The use of semantic information by MAGE's evaluation measure is similar to Crespi-Reghizzi's use of structural information [6] for the inference of context-free grammars.[11] The major difference is that Crespi-Reghizzi includes a complete structural description with each of his examples. MAGE requires analogous information; however, all semantic/syntactic structure is pre-programmed into the

---

[11] Discussed in the second section of this chapter.

domain model of MAGE, and the program itself selects the structural information, which includes meaning and topic in this context, that should be associated with each example.

The use of semantics to construct and evaluate extensions to a grammar is also related to some proposed linguistic models of human language acquisition. The viewpoint that considers MAGE an implementation of these models is discussed in the next chapter. A sample learning session with MAGE is given in Appendix II.

4. Acquisition of Language and Grammatical Extension

4.1 A Model of Language Acquisition

There has always been considerable debate among linguists about the process by which children acquire their native language. Most models represent language learning as an active process of hypothesis-formation and hypothesis-testing: the child continually formulates hypotheses about the language she hears and tests them by attempting to use them to understand speech and to construct her own sentences. The child is not initially presented with the entire language but with a small subset of the vocabulary and syntax which gradually expands as her competence increases [22].

According to a model discussed by Dale [9], a hypothesis is confirmed if it accounts for the data already available and successfully predicts future sentences; otherwise it fails. However, a verdict of success or failure is according to the child's perceptions of language, not an adult's. A grammar that generates the sentence "Shoes on" would be unacceptable to an adult, yet it is considered successful by the two-year-old child who hears "Put your shoes on" as "Shoes on". Hypotheses thus confirmed become part of the evolving grammar used by the child. This grammar is *descriptively adequate*, which means it makes 'accurate' predictions about the correctness or deviance of sentences that the child has never heard before, as well as being *observationally adequate*, which means it accounts for all the sentences that have already been heard.

According to a similar model developed by Chomsky [4, 5], not only does the

observationally adequate grammar account for the observed sentences in the sense of recognizing their structural organization, but also it makes it possible for the child to *understand* the meaning of these sentences. Likewise, the descriptively adequate grammar is capable of understanding infinitely many sentences that the child has never heard.

This model makes the controversial proposition that the child may *know* about certain aspects of language: some knowledge is innate, and the child need not learn these aspects in the usual sense. These innate aspects of language are called the *universal grammar* and, according to the model, form the basis for hypothesis formation and evaluation.

Chomsky's model is founded in the *rationalist* school of linguistic thought, which states that the structure of language is to a considerable degree specified biologically, and the function of experience is to activate this innate capacity and turn it into linguistic competence [4]. The rationalist claims that a great deal of psychological structure is innate and that the human child has a specific, and strong, capacity for language. These ideas are supported by the species-specific and species-uniform attributes of language, i.e. all humans and only humans use language, and by the surprisingly small degree of difficulty a child has with the general mechanisms of language: the notion of a sentence, the establishment of word classes and rules for combining them, and so forth.

The rationalist theory postulates the existence of a universal grammar, such that a successful model of a universal grammar would include exactly those features of language that children do not have to learn and would exclude all the unique features of their particular languages that children must acquire from the speech

they hear. It is a system of principles that categorizes the class of possible grammars by specifying how particular grammars are organized, how the different rules of these components are constructed, and how they interact.

The theory proposes two types of universal features: *substantive* and *formal*. The set of substantive rules includes claims that items of a particular kind in any language must be drawn from a fixed class of items. The formal linguistic universals include more abstract conditions involving the character of the rules that appear in grammars, conditions imposed on these rules, and the ways in which they are interconnected. For example, every human language utilizes the same basic grammatical categories (substantive) -- sentences, noun phrases, verb phrases, etc. -- and uses the same grammatical relations among these categories (formal) -- subject and predicate, verb and object, etc. [18].

According to these models, the task of the child acquiring a language is to choose from among those grammars allowed by the principles of universal grammar that grammar that is compatible with the limited and imperfect data available to her. The child is faced with a finite set of utterances, many of them ungrammatical (due to slips of the tongue, false starts, memory lapses, etc.), that she has heard from her parents and other people in her environment. From these utterances, she must deduce the underlying rules in order to use her language.

The concept of a restrictive, universal mold for grammatical development is supported by the similarities observed by Dale [9] between the early speech of children in different cultures learning widely divergent languages. According to his observations, a child's earliest grammar usually includes a two-word syntactic structure with two classes of words, *pivot* and *open*. The pivot class is small and

each word in it is used with many different words from the much larger open class. For example, an English-speaking child might say "bandage on", "blanket on", "fix on", etc. For this child *on* is a pivot word; it is always used in the second position and many other words can occur with it. Or the child might say "allgone shoe", "allgone lettuce", "allgone outside", and others; here "allgone" is a pivot that always occurs in the first position. A pivot word may be the first or the second element in two-word utterances, but each pivot word has its own fixed position.

As the child grows older and has more experience with her language, she begins to use three-word sentences that are simply pivot-open sentences with an additional word. Agent-object and agent-action constructions merge into the more complex but more meaningful agent-action-object construct. Eventually the child develops the concepts of noun phrase, verb phrase, and all the other complex syntactic structures of the English (or other natural) language.

Although MAGE borrows from these theories of language acquisition and universal grammar, this report is not related to the controversy surrounding these models and rationalist theory in general. The author does not intend the analogy between MAGE and these models (presented below) as an endorsement of any linguistic theories; the analogy is provided as a vehicle for putting in perspective the mechanisms used by MAGE. It may be useful to consider MAGE an implementation of some aspects of these models.

Although an ATN grammar comprises a large portion of MAGE's 'universal grammar', the author does not believe that the augmented transition network formalism is in any way related to the internal organization of the child's grammar. Dresher and Hornstein [10] describe the claims of some linguists that experimental

evidence supports the view that the ATN model is a psychologically realistic model of certain aspects of human linguistic comprehensions; Dresher, Hornstein, and many other linguists disagree. This report is not related to these debates.

Throughout the rest of this chapter, the term 'the child' refers to the human language acquisition mechanisms postulated in the proposed linguistic models discussed above. The author does not claim that the grammatical extension process implemented as MAGE is in any way related to *real* children, or the unknown processes through which they learn language.

4.2 The 'Universal Grammar' of MAGE

Several aspects of these models are 'implemented' as components of the grammatical extension machine: MAGE forms hypotheses that attempt to account for the data it receives. The hypotheses are derived from the program 'universal grammar', which consists of knowledge of the domain and the properties of the grammar it is extending. The kinds of hypotheses that MAGE can formulate are constrained by the set of general extension models, or 'universal rules', presented in Section 3.3. MAGE tests each hypothesis by determining whether it is adequate to 'understand' the example that motivated it. If a hypothesis is inadequate, another hypothesis is formulated and tested until the program has found an extension that enables it to parse the example.[12]

The domain knowledge of MAGE is very similar to the model of a universal grammar presented above. Although the program might be presented with samples from any of a variety of 'dialects' of chatter (e.g. ham radio, military, diplomatic,

[12]This process is described in detail in Sections 3.3 and 3.4.

shipping), the extended grammar will conform to the universals of the radio domain and of the augmented transition network representation for the grammar.

The radio-domain universals include the structural constraints such as network protocol, which limits the types of things that can be 'said' during conversations, and results in the rigid topical breakdown of the ATN into the CONTACT, TRAFFIC, REQ-INFO and END-CNCT subnetworks and the hierarchical organization of these subnetworks into topical subdivisions. These rules are analogous to the formal universals described in section one of this chapter, because they not only constrain, but also define, the character of the grammar.

The Morse code domain also specifies the 'syntax' of q-sign arguments, the existence of 'noise' words, and the internationally defined q-signs. These rules are analogous to the substantive universal rules, which include assertions that structural components and semantic elements must be drawn from prescribed classes.[13]

The built-in ATN also constitutes a set of 'formal universals', which constrain the character of rules that can appear in grammars, since it rigidly defines the type of grammar the program was designed to extend. The ATN model prescribes the types of things that can be stored in registers, what tests and actions can do with registers, and the push and pop (call and return) mechanisms and embedded organization of subnetworks into a transition network grammar.

---

[13]The domain aspects listed here are discussed in depth in Sections 2.3 and 2.4.

4.3 Hypothesis Formation and Evaluation

For each example transmission, MAGE formulates a set of hypotheses for extending the syntactic/semantic structure of the ATN, plus a set of hypotheses for adding function specifications to extract the meaningful content of the example. The mechanisms used here are similar to the linguistic models described in the first section of this chapter. According to those models, the rules formulated by the child must meet the universal conditions imposed on the character of grammatical rules; likewise, MAGE is limited to the forms provided by the set of models illustrated in Section 3.3. Neither the 'model child' nor MAGE is even capable of considering grammatical hypotheses that do not meet their constraints.

The proposed linguistic models predict that the child will ignore sentences whose structure and/or vocabulary are too unusual, too different from what she already knows; MAGE returns a verdict of 'unlearnable' every time it receives a difficult example, until it has acquired enough vocabulary and contextual structure to simplify the learning of this example to the matching of one new phrase to its hypothesis-formation models. Both MAGE and the 'model child' learn by a bootstrap process. As MAGE is exposed to more and more example transmissions, the conversations it can parse become more complex.

The core grammar of the grammatical-extension machine is similar to the pivot grammar discussed by Dale [9], in that most chatter phrases revolve around one 'pivot' word, often a q-sign, that determines the meaning of the other words. The ability to associate pivot words with only one or two potential subnetworks, coupled with the ease with which most pivot words are recognized (e.g. all q-signs begin with the letter 'Q'), is probably the most important feature of MAGE's evaluation measure.

Without this ability, the selection mechanism would probably have to rummage through each of the thirteen subnetworks, possibly during several passes, to find the 'best fit' for each example.

The model of a hypothesis-selection mechanism proposed by Chomsky [4, 5] and discussed further by Dale [9], which would accept only those hypotheses that make it possible for the 'model child' to make sense of an utterance, according to her perception of 'sense', is analogous to the evaluation measure utilized by MAGE: a hypothesis is accepted only if it provides a parser with the ability to understand the example transmission. Extensions to the grammar are made in such a way that learning one new sentence actually results in the power to understand arbitrarily many new sentences, since many paths through the ATN may follow the new transitions. Thus the resulting grammar is descriptively adequate; theorists claim that a human grammar developed according to their models would also be descriptively adequate.

MAGE does not use any of the particular universal rules postulated by linguists attempting to explain the very complex processes of language acquisition by children, nor does it copy the specific tenets of any of the theorized universal grammars (no one knows exactly what the universal grammar used by children actually consists of, or even whether it really exists). What MAGE does do is implement the *concept* of a universal grammar, with universal rules that severely constrain the development of a grammar that accepts the particular dialect of chatter being learned. MAGE also implements the idea of selecting only those hypotheses that provide an accurate mechanism for 'understanding' -- or in this case extracting the important information from -- the motivating example(s).

5. MAGE: A Learning System

5.1 A Model for Learning Systems

The organization of the computer program was strongly influenced by the research described by Smith et al. [20], and MAGE conforms closely to their model of a learning system. The model details the functional components felt to be essential for any learning system, independent of the techniques used for its construction and the specific environment in which it operates.

Smith et al. define a learning system as "any system which uses information obtained during one interaction with its environment to improve its performance during future interactions". The performance of MAGE complies with this definition, as any examples that are added to the grammar's understanding capability are also used by the bootstrap process to extend the grammar to accept future examples.

The learning system model proposed by Smith consists of six elements. The Instance Selector selects suitable training instances from the environment. The Performance Element generates an output in response to a training instance. The Critic analyzes the output of the performance element in terms of some standard of performance. The Learning Element makes specific changes to the system in response to the analysis of the critic. The Blackboard contains system information, e.g. the emerging knowledge base, that is used by all functional components. Finally, the World Model contains the general assumptions and methods that constrain system activity.

My experience with MAGE conformed to this model in one additional way: as designer, I viewed the entire learning system as a program whose performance

needs improvement, and I selected instances, criticized performance, and made changes accordingly. In other words, the designer's activities can be modeled by a system whose components are identical to those described above. This leads to the interesting concept of layered learning systems, each higher layer able to change the world model (vocabulary, assumptions, etc.) of the next lower layer on the basis of criticizing its performance on a chosen set of instances.

5.2 MAGE Components

5.2.1 Instance Selector and Blackboard

The Instance Selector performs the trivial operation of accepting whatever example the user provides and transforming it to the proper data structure for system manipulation. It may request the user to answer certain questions about the current example. For example, if the current example were "VVV ROCK DE SALT QSA ? K" ("Rock, this is Salt. What is the strength of my signals? Over"), the Instance Selector would look up each word in the vocabulary list of the World Model and find that "VVV" is an unknown word. It would print: *'VVV' IS AN UNKNOWN WORD. DOES IT HAVE A SYNONYM ON THE FOLLOWING LIST?* (followed by the list). The operation of MAGE on this example is described in Appendix II. The Instance Selector provides half of the user-program interface.

The other half of the user-program interface is the Blackboard, which prints statements about each extension the program makes to the grammar, e.g.

[Adding new transition 'VVV' from state 0 to 0 of ID-OP]

(the result of the above example). In addition, all communication between modules is considered part of the Blackboard. Most communication takes place via standard

passing of arguments, and use of the same variables when parts of one module are embedded inside another. There are also some global variables that designate what portions of the grammar have been altered during this learning session and other dynamic information.

5.2.2 World Model

The World Model contains the universal grammar,[14] which includes all knowledge MAGE has about the Morse code radio network domain. The core grammar is considered a component of the World Model. It contains the subnetworks diagrammed in Appendix III, but not any alterations that have been made during the current learning session: these belong to the Performance Element. The World Model has some concise, hand-gathered collections of informational items that are distributed throughout the core grammar and would be difficult to find without these indices, e.g. the set of all subnetworks and symbols that can immediately follow any terminal state in the **QUAL-CNCT** subnetwork.

The World Model also includes a set of specifications for the tests and actions. A 'specification' describes in what circumstances the test or action should be associated with a transition and what arguments should be passed to the pre-coded function that implements the test or action.

The spellings of sixty q-signs are known *a priori* by the system. Each q-sign is associated with one or more topical subnetworks and a possible argument syntax. However, only five of the sixty q-signs appear on transitions in the core grammar, and MAGE must receive at least one example for each of the other q-signs in order

[14] Section 4.2

to understand conversations containing that q-sign. A synonym table, which includes all q-signs and all other vocabulary contained in the core grammar, is maintained.

5.2.3 Performance Element

The Performance Element consists of two components: a TN parser and the current version of the ATN grammar. The TN parser is based on the ATN parser of CATNIP [16], but it does not save nor use any contextual information, since it is only trying to accept a sentence or conversation rather than trying to comprehend it. It doesn't need tests to determine which words should be accepted by a transition with the symbol "any" because all code-groups and English words are written as "any" in the example. There is no reason that MAGE needs to recognize code-groups and English words as such, since this task is successfully performed by COMDEC [21].

Although there is only a single data structure implementing the ATN grammar, the core grammar is said to be an element of the World Model, and the current version of the grammar (i.e. the core grammar plus various extensions depending on the history of the current learning session) is considered a component of the Performance Element. The current state of the non-q-sign vocabulary is also part of this element, while the original vocabulary is part of the World Model. This conforms to Smith's model of a learning process as operating on or making changes in the Performance Element, where only the designer can alter the World Model.

When the Performance Element is processing an example, it reaches one or more states where none of the transitions leaving those states matches the next word in the example (unless the example is already accepted by the grammar). When this occurs, it passes a set of pointers to these states and a pointer to the next word

in the example to the portion of the Critic that is embedded in the Performance Element.

5.2.4 Critic and Learning Element

The Critic performs three semi-independent functions: Evaluator, Diagnostician, and Therapist.

As Evaluator, it evaluates the Performance Element's ability to parse each example and 'tells' the parser to halt when the Critic realizes that the parser cannot understand the next phrase of the example. The Evaluator is embedded in the Performance Element. As described above, when the parser halts it provides the Critic with the necessary state information to perform its hypothesis-formation task.

As Diagnostician, the Critic localizes the reasons for poor performance by noting at which state(s) the parser was forced to halt. It enumerates a set of hypotheses based on the structural match between the example and the localized position in the grammar.[15]

In Therapist mode, the Critic performs the evaluation measure.[16] It selects one of the hypotheses formulated while in Diagnostician mode, and returns to Diagnostician mode. The Diagnostician enumerates a set of test/action specifications, and the Therapist selects some of these to augment the transitions in the newly chosen structural extension.

The Critic passes the chosen structural and test/action hypotheses to the Learning Element, which utilizes knowledge of implementation details to determine

[15] Section 3.3

[16] Sections 3.4 and 4.3

how to alter the ATN data structure to include the current extension. Actually, the term 'Learning Element' may be a poor choice for this module since it simply makes the changes suggested by the Critic; however, Smith et al. [20] describes the 'learning process' in as simply an addition of already formulated and selected rules to permanent memory.

5.3 Implementation Details

The MAGE subsystem is implemented in MDL ('Muddle') [15] and runs on a Digital Equipment Corporation KA-10 under the ITS operating system. MAGE includes about 1300 lines of MDL code, and the compiled version requires about 47 blocks of memory beyond the MDL interpreter. (A block contains 1024 36-bit words.)

6. Conclusions

6.1 Capabilities and Limitations

This report describes the development of a computer program, MAGE, that acquires and organizes much of the domain-specific knowledge required by the related system, CATNIP [16], to process conversations over Morse code radio networks. MAGE incorporates several of the levels of learning ability described by Winston [23]. On the lowest level, it 'learns' the domain-specific knowledge contained in its core grammar by being programmed. On higher levels, it receives additional information by being told, in the language of the domain rather than a programming language, and it acquires the rest of its domain-specific knowledge via learning by example. It is not able to learn by discovery.

MAGE uses the parser's ATN knowledge base as a 'core' on which it builds the developing grammar. The core contains a certain amount of domain knowledge that was readily available to the human who developed CATNIP and MAGE but could not be acquired by the present version of MAGE. The inclusion of a core knowledge base represents learning by being programmed. The core includes:

- the discourse structure imposed on conversations by radio-network protocol
- the types of information conveyed during Morse code conversations
- the set of generic tokens and information about how to narrow down what should and should not be matched by these tokens
- the spellings and meanings of the internationally defined q-signs
- the syntax of a few basic phrases and the meanings of the words that appear in these phrases

- the knowledge that 'noise' words exist
- how to format the various types of information for human-readable output

This knowledge is reflected in the core as:

- the top-down organization of the ATN knowledge base into thirteen semantically categorized subnetworks
- the internal structure of the core subnetworks
- the registers, tests, and actions
- a lexicon that associates the q-signs and other words contained in the core vocabulary with their synonyms, if any, among the known words
- the printing functions

MAGE receives as input individual transmissions, each containing either no new information or exactly one new phrase. In some cases where the example contains unknown words, MAGE must ask the user for additional information about the new words, and the user responds in the language of the domain rather than by additional programming: this is learning by being told.

MAGE derives enough information from each example to extend the knowledge base to process the new phrase in the context of the example transmission and related contexts. The new extension becomes an integral part of the grammar, utilized henceforth by CATNIP -- to select the correct transcription of a conversation from among the many transcriptions suggested by COMDEC and to produce a human-readable summary of the information conveyed during the conversation -- and by MAGE -- to aid in the bootstrap procedure that extends the grammar. This process represents learning by example. The procedure followed by MAGE is:

1. MAGE attempts to parse the example transmission using the current version of the ATN.

    a. If the example can already be parsed, get a new new example.

    b. Otherwise, the parse failed at some particular word in the example sentence; that is, it could not advance any of the one or more parse paths by another transition matching this word. Call the last state in each failed path a 'last-matched-state'. Call the word on which the parse failed the 'next-word'.

2. MAGE looks for some word following the next-word that follows the end of the new phrase.

    a. This word and all words following this word in the example match some connected sequence of states and transitions in the ATN that can be reached, via existing transitions, from one or more of the last-matched-states. Call the first state in each such sequence an 'end-of-phrase'.

    b. Or, there is no such word and the new phrase ends at the end of the transmission: the extension representing the new phrase must end in a terminal state, also called 'end-of-phrase'.

3. MAGE compares each last-matched-state/end-of-phrase pair to the set of models, where any of the three states corresponding to those in Model 0 may match the last-matched-state and any terminal state may match the end-of-phrase, depending on the particular model and circumstances.

    a. It finds one or more models for each pair that could be used to construct an extension for the new phrase. Call each such model and pair combination a 'template'.

    b. It selects the best template on the basis of a set of heuristics and constructs the structural

component of an addition to the ATN, called an 'extension'.

4. MAGE selects a set of test specifications and a set of action specifications for each of the transitions in the extension.

a. The specifications are chosen according to a set of heuristics that consider the transition symbol, the context of the rest of the transmission, and the particular subnetwork to which the extension was made.

b. MAGE adds the specifications to the previously constructed extension and gets a new example.

MAGE may extend the knowledge base to include an arbitrarily large number of new phrases for discussing the concepts allowed by the known discourse structure. It augments the transitions that process the words of these phrases with tests that provide filters for generic tokens and actions that extract the information from a phrase that provides temporary context and contents for the summary output.

MAGE may be considered an implementation of some linguistic models of human language acquisition proposed by Chomsky [4, 5]. This analogy is very natural, since language acquisition seems very closely related to grammatical extension.

- The domain-specific knowledge contained in the core knowledge base corresponds to the innate 'universal grammar'.
- The example transmissions correspond to the utterances heard by the 'child'.
- The models and associated heuristics correspond to the 'universal rules'.
- The creation of several templates and consideration of possible test/action specifications corresponds to the formation of competing

hypotheses.

- The construction of one extension that processes the example corresponds to the selection of one hypothesis that adequately explains the data.

Even if these models turn out to be poor descriptions of the learning processes actually used by children acquiring their native language, this research has demonstrated that these theories are still useful in the design of computer programs that successfully learn by example.

However, MAGE has many limitations:

- It is not able to recognize changes to the discourse structure or to the type of information conveyed during conversations, should these occur. In other words, it cannot create new subnetworks, registers, tests, or actions, nor discard existing ones.

- It includes no mechanism for automatically adding the meanings of new q-signs or other vocabulary words, unless these words are synonyms of previously known words; however, this can be easily programmed by a human.

- It assumes the existence of an intelligent and knowledgeable user, who does not simply type in complete new transcripts but rather edits the example transmissions so that they each include only one new phrase. This means the user should have some knowledge of the current capabilities of the knowledge base. Fortunately, MAGE performs adequately most of the time with a naive user, except where the transcript includes a large number of 'interruptions'.

- Most notably, the current version of MAGE can not deal with the interruption problem and is able neither to extend the Interrupt Subnetwork and related lower-level subnetworks nor filter out interruptions from example transmissions.

These limitations are what separate grammatical or knowledge-base extension from grammatical inference. If MAGE could do all these things, it would be able to acquire, from transcripts, all the domain-specific knowledge required by CATNIP.

That is, it could learn by discovery, the highest and least understood form of learning.

A system that could do all the things listed above, without prior domain-specific knowledge, could automatically acquire the particular domain-specific knowledge required by any system whose knowledge base could be derived by a human from a reasonable amount of data taken directly from the domain and organized as an augmented transition network. It would be a solution to the very difficult problem of grammatical inference of a context-sensitive grammar.

6.2 Suggestions for Future Research

The research described in this report represents a small step in the development of a grammatical-inference machine that could construct the knowledge base or grammar necessary to parse a natural language from scratch, i.e. without requiring a programmer-defined organization of subnetworks, registers, tests, and actions. The design of this machine would require the removal of all the limitations described above, which involves finding the solution to two major artificial-intelligence problems.

One of the problems to be solved is grammatical inference of the transition network component of the ATN from an incomplete set of examples, each containing an arbitrary amount of new information and an arbitrary amount of old information. The current state of machine inference of context-free grammars, which are equivalent to non-augmented transition networks, assumes a *structurally complete* sample set. However, it is impossible to put together a sample set using every production or rule in a grammar when it has not yet been agreed what all the rules are for any natural language. Therefore, either a new inference algorithm with

different assumptions or a completely different method for deriving grammars for natural languages would have to be developed.

The solution to the other problem requires the automation of both the process of recognizing the need for certain registers, and the process of writing algorithms, or abstract function descriptions, for the tests and actions. Once an algorithm has been generated in some simple 'programming language' known by the learning program, a human programmer could code the tests and actions in the actual language (e.g. Pascal, PL/1, MDL) suitable for the particular environment.

Both problems might be considerably more tractable if restricted to Morse code or some equally simple domain, and if they could be solved independently. That is, the ability to utilize the domain-specific knowledge inherent in a programmed version of one of the two components may make it easier to develop an automatic mechanism to perform the other function.

For example, a grammatical-inference machine might use some domain knowledge, such as the topic of q-signs or the type of information conveyed during conversations, to develop the set of subnetworks for processing Morse code conversations. The Morse code domain simplifies the test/action problem by restricting the potential contents of registers to words and phrases selected from transmissions. Tests are restricted to putting additional constraints on generic tokens by comparing the contents of registers to the current word(s); actions are restricted to selecting/storing important information and deleting information that is no longer desired. This knowledge might be utilized by a program that automatically generates registers, tests, and actions.

Regardless of whether these problems are ever dealt with for the specific case

of automatic generation of the knowledge base for parsing Morse code conversations, it is hoped that they will someday be solved for the general case, so that machine acquisition of natural language will become possible.

References

[1] Alfred V. Aho and Jeffrey D. Ullman.
The Theory of Parsing, Translation and Compiling -- Volume I: Compiling.
Prentice-Hall, Inc., 1972.

[2] The American Radio Relay League.
The Radio Amateur's Operating Manual.
ARRL, Inc., 1972.

[3] A.W. Biermann and Jerome A. Feldman.
On the Synthesis of Finite-State Acceptors.
Technical Report AIM-114, Stanford Artificial Intelligence Project, April, 1970.

[4] Noam Chomsky.
Aspects of the Theory of Syntax.
The M.I.T. Press, 1965.

[5] Noam Chomsky.
Language and Responsibility.
Pantheon Books, 1977.

[6] S.M. Chou and King-Sun Fu.
Inference for Transition Network Grammars.
In *Third International Joint Conference on Pattern Recognition*, pages 79-84.
November, 1976.

[7] Craig M. Cook.
Experiments in Grammatical Inference.
Technical Report TR-257, University of Maryland Computer Science Center,
August, 1973.

[8] Stefano Crespi-Reghizzi.
Reduction of Enumeration in Grammar Acquisition.
In *Second International Joint Artificial Intelligence Conference*, pages
546-52. 1971.

[9] Philip S. Dale.
Language Development: Structure and Function.
The Dryden Press, Inc., 1972.

[10] B. Elan Dresher and Norbert Hornstein.
On Some Supposed Contributions of Artificial Intelligence to the Scientific
Study of Language.
Cognition 4:378-97, 1976.

[11] B.M. Eisenstadt, B. Gold, D.M. Nelson, T.S. Pitcher and O.G. Selfridge.
MAUDE.
Technical Report 24-57, Lincoln Laboratory Group Report, 1957.

[12] Jerome A. Feldman, James Gips, James J. Horning and Stephen Reder.
Grammatical Complexity and Inference.
Technical Report TR-CS-125, Stanford Artificial Intelligence Project, June, 1969.

[13] Victoria Fromkin and and Robert Rodman.
An Introduction to Language.
Holt, Rinehart & Winston, 1978.

[14] King-Sun Fu and Taylor L. Booth.
Grammatical Inference: Introduction and Survey -- Part I.
I.E.E.E. Transactions on Systems, Man, and Cybernetics SMC-5(1):95-111, January, 1975.

[15] S.W. Galley and Greg Pfister.
The MDL Programming Language.
M.I.T. Laboratory for Computer Science, 1979.

[16] Gail E. Kaiser, David Sherry, and Albert Vezza.
Understanding Morse Code Conversations Using an Augmented Transition Network Grammar.
Technical Report SYS 17.06, M.I.T. Laboratory for Computer Science, Programming Technology Division, July, 1979.

[17] Bruce and Kathleen Knobe.
An Algorithm for Inferring Context-free Grammars.
Technical Report TR-73-7, The Hebrew University of Jerusalem, Dept. of Computer Science, December, 1973.

[18] David McNeill.
The Acquisition of Language: The Study of Developmental Psycholinguistics.
Harper & Row, Publishers, 1970.

[19] Graeme D. Ritchie.
Augmented Transition Network Grammars and Semantic Processing.
Technical Report CSR-20-78, University of Edinburgh, Dept. of Computer Science, January, 1978.

[20] Reid G. Smith, Tom M. Mitchell, Richard A. Chestnek, and Bruce G. Buchanan.
A Model for Learning Systems.

In *Fifth International Joint Conference on Artificial Intelligence*, pages 338-43. August, 1977.

[21] Albert Vezza, P. David Lebling, Edward H. Black, Timothy A. Anderson, John F. Haverty, David Sherry, and Gail E. Kaiser.
Machine Recognition and Understanding of Manual Morse.
In *Distributed Sensor Nets*, pages 125-36. Proceedings of a Workshop held at Carnegie-Mellon University, December, 1978.

[22] R. Michael Wharton.
Grammatical Inference and Approximation.
Technical Report TR-51, University of Toronto Dept. of Computer Science, February, 1973.

[23] Patrick Henry Winston.
Artificial Intelligence.
Addison-Wesley Publishing Co., 1977.

[24] William A. Woods.
Transition Network Grammars for Natural Language Analysis.
Communications of the ACM 13(10):591-606, October, 1970.

I. A Morse Code Conversation

A typical example of a goal-oriented Morse code conversation is given below, with each transmission followed by an English transcription. 'ROCK' and 'SALT' are two operators. Very little of this conversation can be understood by the parser using only MAGE's core grammar, which is presented in Appendix III, although all of it can be parsed using the complete grammar actually used by CATNIP. However, MAGE is capable of extending the grammar so that the parser can 'understand' this entire conversation. The sample learning session presented in Appendix II shows how MAGE extends the core grammar to understand new transmissions; many of the transmissions in this conversation are used as examples.

VVV VVV ROCK ROCK ROCK DE SALT SALT QSA ? K

("[Hey] Rock, this is Salt. What is the strength of my signals?
Over")

VVV VVV ROCK DE SALT QSA ? QRK ? QSA ? QRK ? QTC QTC K

("[Hey] Rock, this is Salt. What is the strength of my signals?
What is the intelligibility of my signals? [Can you hear me?]
I have messages for you. Over")

SALT DE ROCK QSA 5 QRK 5 GA K

("Salt, this is Rock. The strength of your signals is very good. The
intelligibility of your signals is excellent. [I can hear you!]
Go ahead. Over")

HR TFC HR TFC OK ? K

("Here's some traffic. [I'm going to send a message now.] Okay?
Over")

QRV K

("I am ready. Over")

NR 1 GR 200 1500 BT <100 code-groups> BT BT <100 code-groups> BT QSL ? K

("[Message] Number one, with 200 groups, at 1500 hours (3 p.m.) break *<100 code-groups>* break *<100 code-groups>* break. Can you acknowledge receipt? Over")

N N PSE RPT GRPS 25 , 40 , 98 K

("No. Please repeat groups 25, 40, and 98. Over")

OK OK GRP 25 <code-group> / GRP 40 <code-group> / GRP 98 <code-group> K

("Okay. Group 25 is *<code-group>*. Group 40 is *<code-group>*. Group 98 is *<code-group>*. Over")

TKS QSL UR MSG NR 1 NW K

("Thanks. I am acknowledging receipt of your message number one now. Over")

QTC ? K

("Do you have any messages for me? Over")

QRU QRX ? K

("I have nothing for you. When will you call me again? Over")

QRX NXT TMW OK ? K

("I will call you again tomorrow. Okay? Over")

C C SK SK

("Okay. End of contact")

VA

("End of contact")

II. A Learning Session with MAGE

An example of MAGE's performance is given below for each of the seven general models presented in Section 3.3. The prose in brackets is that printed by MAGE for the given example. In each case, Figure a shows the model selected by the hypothesis-formation algorithm; Figure b displays the original subnetwork selected by the evaluation measure; and Figure c gives the result of applying the model to the example and the chosen subnetwork. Since it is difficult to show tests and actions in the diagrams, the selected test/action specifications are presented in the brief discussion below each example.

Example 1

ROCK DE SALT PSE ANS QTC K

[Changing state 1 of TFC-INFO to TERMINAL]

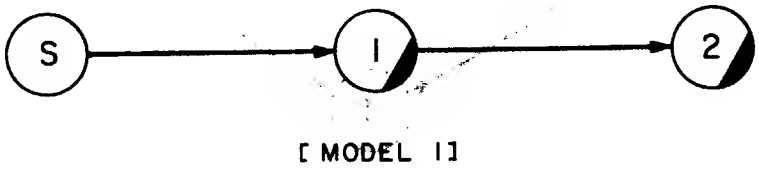


Figure 10a: Model 1

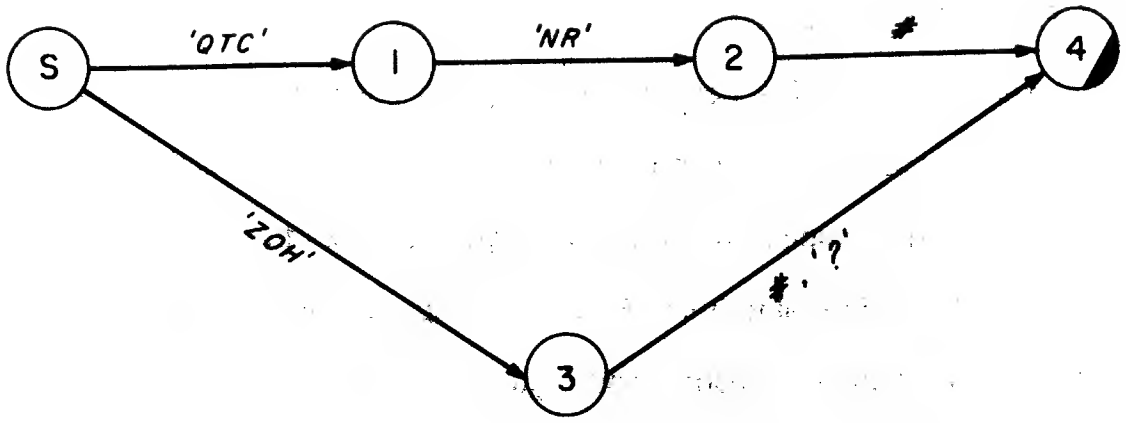


Figure 10b: Core **TFC-INFO**

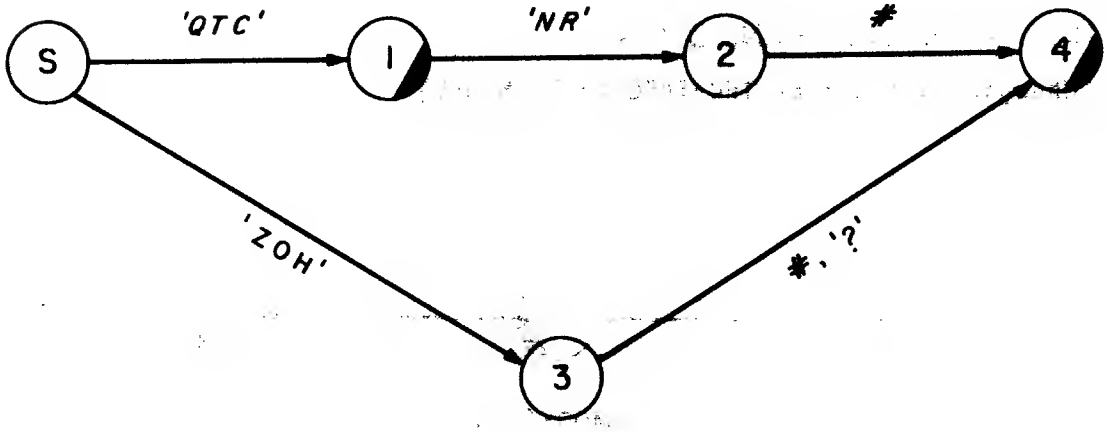


Figure 10c: Extended **TFC-INFO**

("Rock, this is Salt. Please answer, I have messages for you. Over")

The phrase "ROCK DE SALT" is accepted by the **ID-OP** subnetwork (Figure 19), and "PSE ANS" is accepted by the **QUAL-CNCT** subnetwork (Fig. 21). When a phrase accepted by **ID-OP** is followed by a phrase accepted by **QUAL-CNCT**, the two phrases together are accepted by the **CONTACT** subnetwork (Fig. 18). This subnetwork may be followed by the **TRAFFIC** subnetwork (Fig. 22), as well as by

another occurrence of **CONTACT**, as shown in the **OVERALL** subnetwork (Fig. 17), the highest level subnetwork in this ATN. "QTC" matches the symbol on the first transition of **TFC-INFO** (Fig. 23), which is pushed to (called) by the first transition of the higher-level **TRAFFIC** subnetwork. However, "K" does not match the next transition in **TFC-INFO**; instead, it matches the transition following **TFC-INFO** in **TRAFFIC**. This indicates that the next-state of the "QTC" transition should be a terminal state so it can pop (return) to **TRAFFIC**, so MAGE changes it.

Since no transitions are added, it is not necessary for MAGE to consider adding new tests or actions.

Example 2

QSL MSG NR 3 ? K

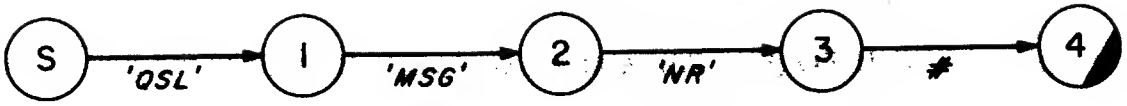
[Adding new transition '?' to state 4 of ACKNOW]

[Also adding 1 new states to ACKNOW]

[States: TERMINAL]

[MODEL 2]

Figure 11a: model 2



[CORE ACKNOW]

Figure 11b: core **ACKNOW**

[EXTENDED ACKNOW]

Figure 11c: extended **ACKNOW**

("Can you acknowledge receipt of message number three? Over")

The phrase "QSL MSG NR 3" is accepted by the **ACKNOW** subnetwork (Fig. 28) and "K" matches the symbol on the transition following a (call) push to **ACKNOW** in the higher-level **REQ-INFO** subnetwork (Fig. 26). Since it is known *a priori* that extensions should be made to lower-level rather than higher-level subnetworks whenever possible, MAGE adds a transition "?" to the terminal state of **ACKNOW** and creates a new terminal state that pops (returns) to **REQ-INFO.**

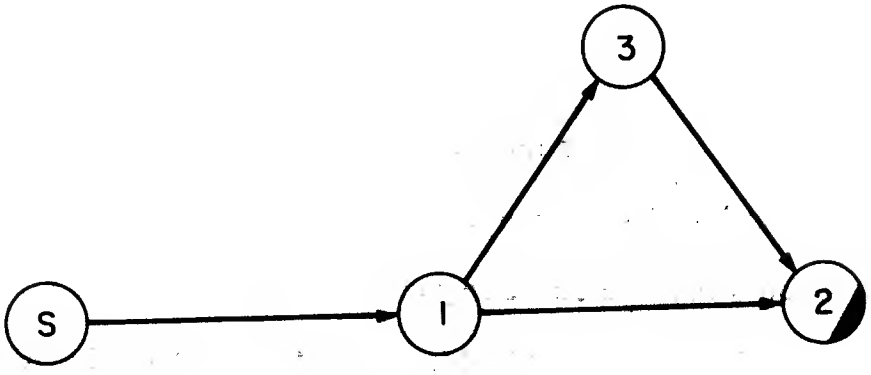
Now the action [SCRATCH input] (store input token in *⟨scratch-pad⟩* register, destroying the previous contents) is already associated with "QSL". Since "?" refers back to the q-sign, the action [Q-PEND SCRATCH] (the token in *⟨scratch-pad⟩* was used as a question; put it in the *⟨pending-question⟩* register of the receiving

operator) is associated with the new transition.[17]

Example 3

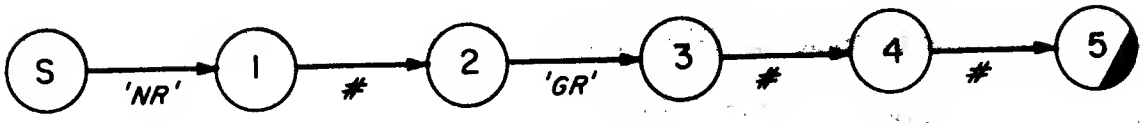
NR 1 GR 200 QTR 1400 any BT QSL ? K

```
[Adding new transition 'QTR' to state 4 of HEADER]

[Also adding 1 new states to HEADER]

[States: # , to TERMINAL]
```



[MODEL 3]

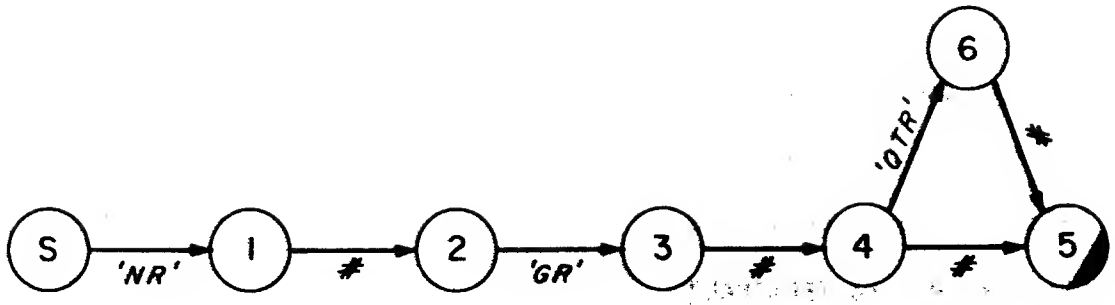
Figure 12a: Model 3



[CORE HEADER]

Figure 12b: Core HEADER

[17] All tests and actions are defined in Appendix III.



[EXTENDED HEADER]

Figure 12c: Extended **HEADER**

("[Now sending message] number one, with 200 groups, at the time 1400 hours. Break *<code-groups>* break. Can you acknowledge receipt? Over")

"NR 1 GR 200" matches the first few transitions of the **HEADER** subnetwork (Fig. 24) and is followed by a transition matching "1400" (i.e. the symbol on this transition is " # "). "any BT QSL ? K" is accepted by the **MESSAG** subnetwork (Fig. 25), which follows **HEADER** in the higher-level **TRAFFIC** subnetwork (Fig. 22). Thus "QTR # " appears to be an alternate way of phrasing this last " # ", so MAGE creates two new transitions "QTR" and " # ", with a new state between them, in parallel with the original transition for " # ".

Since "QTR" is a q-sign followed by an argument, the action [SCRATCH input] is associated with "QTR" and the actions [Q-VAL input] and [Q-ACT SCRATCH] are associated with the argument. [SCRATCH input] stores the input token in the *<scratch-pad>* register, destroying the previous contents; [Q-VAL input] adds the next input token to the *<scratch-pad>* register without destroying the previous contents; [Q-ACT SCRATCH] removes the q-sign and its argument(s) from

<scratch-pad>, determines which register to put them in, and puts them there. The possible registers include *<expected-actions>*, *<quality-of-contact>*, *<general-situation-description>*. In addition, since "QTR #" is another way of phrasing the "#", any tests or actions on the original transition must be copied to the new ones: therefore, [GMT-TIME input] is also associated with the new transition for "#". The action [GMT-TIME input] puts the input token, indicating time of transmission, in the *<time-and-date>* register.

Example 4

VVV ROCK DE SALT QSA ? K

'VVV' IS AN UNKNOWN WORD. DOES IT HAVE A SYNONYM ON THE FOLLOWING LIST?

<list of known vocabulary words that are not q-signs or call-signs>

N

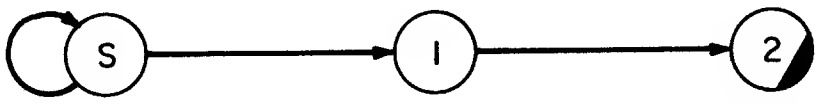
DOES 'VVV' HAVE A QSIGN SYNONYM?

N

COULD 'VVV' BE CONSIDERED A 'NOISE' WORD?

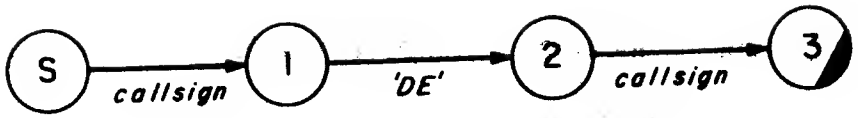
Y

[Adding new transition 'VVV' from state 0 to 0 of ID-OP]



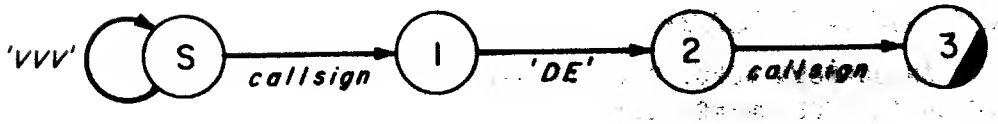
[MODEL 4]

Figure 13a: Model 4



[CORE ID-OP]

Figure 13b: Core **ID-OP**



[EXTENDED ID-OP]

Figure 13c: Extended **ID-OP**

("[Hey] Rock, this is Salt. What is the strength of my signals? Over")

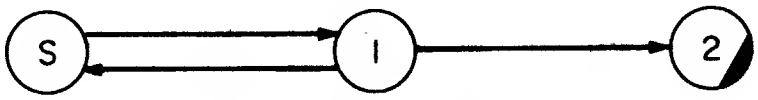
Since "VVV" is a new word, MAGE asks the user to supply some information about its meaning. Since MAGE is told that "VVV" is a 'noise' word, and it is followed by "ROCK DE SALT" which is accepted by the **ID-OP** subnetwork (Fig. 19), MAGE adds a new transition "VVV" as a loop on the start-state of **ID-OP**.

There are no tests or actions associated with noise words.

Example 5

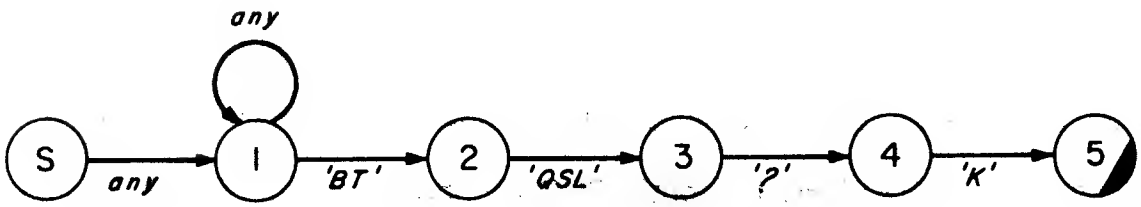
NR 2 GR 150 1600 any BT any BT QSL ? K

```
[Adding new transition 'any' to state 2 of MESSAG]

[The arc has next-state 1]
```



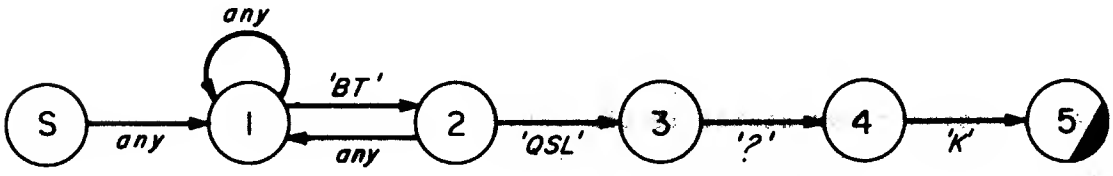
[MODEL 5]

Figure 14a: Model 5



[CORE MESSAG]

Figure 14b: Core **MESSAG**



[EXTENDED MESSAG]

Figure 14c: Extended **MESSAG**

("[Now sending message] number two, with 150 groups, at 1600 hours. Break *<code-groups>* break *<code-groups>* break. Can you acknowledge receipt? Over")

"NR 2 GR 150 1600" is accepted by the **HEADER** subnetwork (Fig. 24), which is followed by the **MESSAG** subnetwork (Fig. 25) in the higher-level **TRAFFIC** subnetwork (Fig. 22). "any BT" is matched by the first two transitions of the **MESSAG** subnetwork, but the second "any" does not match any transitions leaving state 2. Rather than branch to a new path that merges with the old at "QSL", MAGE notes that the second "any BT" also matches the first two transitions of **MESSAG**. MAGE creates a new transition that returns to state 1, so this new phrase can be repeated indefinitely.

The tests and actions that are associated with the original "any" transition from the start-state to state 1 are copied to the new "any" transition: test [GROUP? input] and action [ADD-GROUP input]. [GROUP? input] returns TRUE if the input is probably a code-group or English word; [ADD-GROUP input] increments the *<number-of-words-received-so-far-in-message>* register, and puts the input token in the *<last-word-received-in-message>* register, which is useful for error-recovery.

Example 6

QRX ? K

```
[Adding new transition 'QRX' to state 0 of END-CNCT]
[Also adding 1 new states to END-CNCT]
[States: ? , to TERMINAL]
```

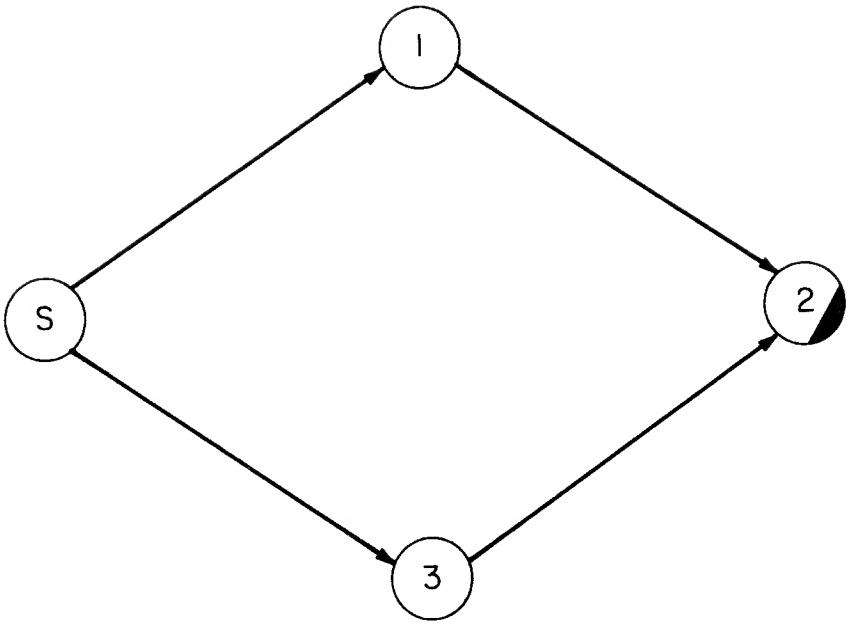


Figure 15a: Model 6

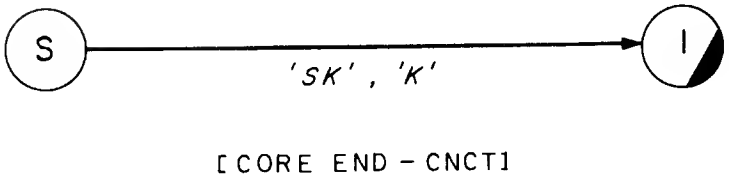


Figure 15b: Core **END-CNCT**

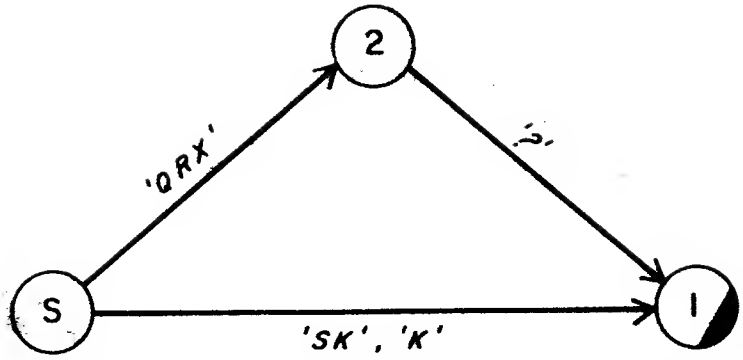


Figure 15c: Extended **END-CNCT**

("When will you call me again? Over")

Here is a situation where the first word of the example doesn't match any transition leaving a start-state. However, the q-sign "QRX" is semantically associated with the **END-CNCT** subnetwork (Fig. 29). Since "K" appears on a transition to a terminal state in **END-CNCT**, and the **END-CNCT** subnetwork can follow itself in the highest-level **OVERALL** subnetwork (Fig. 17), the new phrase "QRX ?" is added to **END-CNCT** as a new path.

Since "QRX" is a q-sign followed by a likely argument, it is associated with the action [SCRATCH input], which saves the q-sign in the *<scratch-pad>* register until its argument(s) are collected. The argument "?" is associated with the action [Q-PEND SCRATCH], which notes that the q-sign found in *<scratch-pad>* was used as a question and stores it in the *<pending-question>* register of the receiving operator.

Example 7

QTC ? K

```
[Adding new transition '?' to state 1 of TFC-INFO]

[Also adding 1 new states to TFC-INFO]

[States: TERMINAL]
```

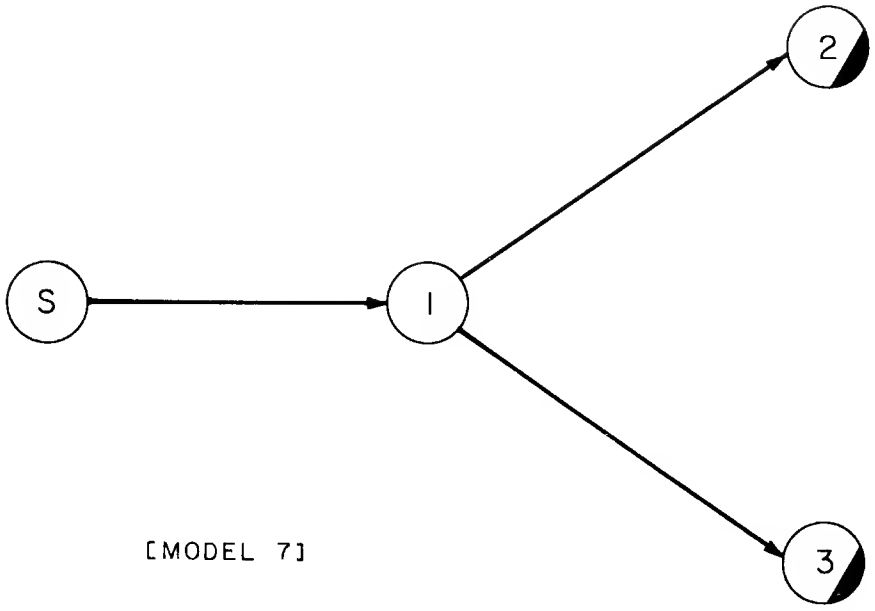


Figure 16a: Model 7

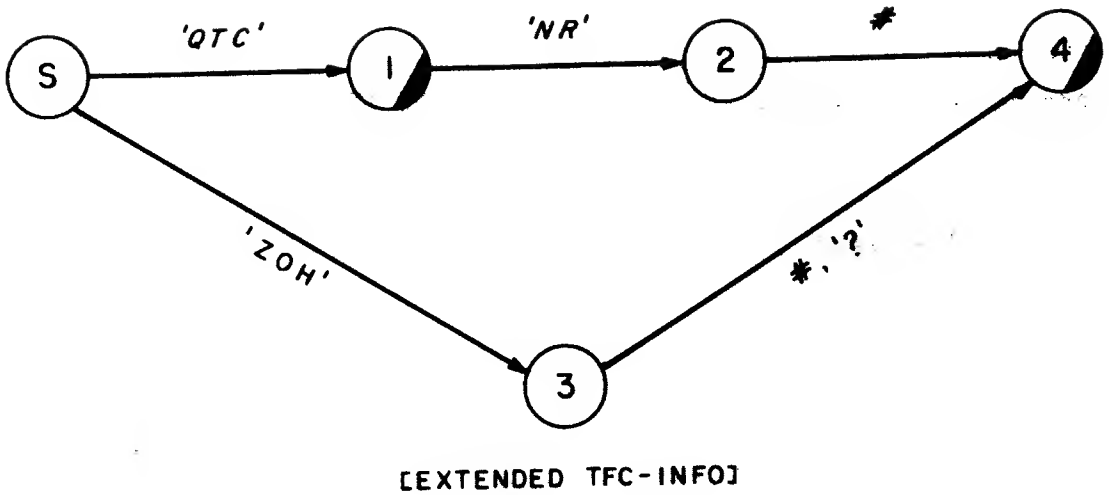


Figure 16b: Recently extended **TFC-INFO** from Figure 10c

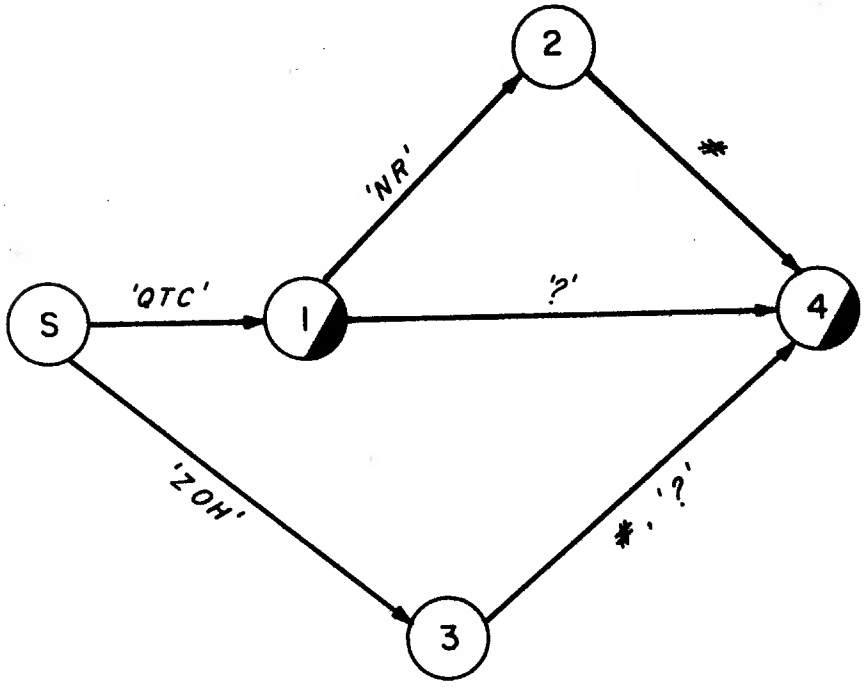


Figure 16c: **TFC-INFO** extended further

("Do you have any messages for me? Over")

In this case, another extension is made to a previously extended subnetwork

(see Figure 10 above). "QTC" matches the first transition in the **TFC-INFO** subnetwork (Fig. 23), but "?" does not match the transition leaving this state, nor does it match any transition leaving the state in the **TRAFFIC** subnetwork (Fig. 22) that can be popped (returned) to from this terminal state. Since "?" is likely to be a q-sign argument, a branch is created in **TFC-INFO** that ends in a new terminal state. (Actually, this terminal state is merged with the other terminal state that has no transitions leaving it in order to minimize complexity.)

Since "?" refers to a q-sign, and [SCRATCH input] is already associated with that q-sign (and will store the token in the *<scratch-pad>* register), the action [Q-PEND SCRATCH] is selected for the new transition (to retrieve the q-sign from the *<scratch-pad>* register and put it in the *<pending-question>* register of the receiving operator).

III. The Core Grammar of MAGE

This appendix includes a list of the chatter words that appear in the core grammar, illustrations of the subnetworks composing the core grammar, a list of registers, and descriptions of the tests and actions. Although the registers, tests, and actions are the same as used by CATNIP[18] [16], the vocabulary and grammar of MAGE are considerably smaller than the grammar used by CATNIP.

Vocabulary

? -- question mark; punctuation and a q-sign argument

-- generic matched by any number

ANS -- "answer"

any -- matches any code-group or English word in message

BT -- "break"; a pro-sign

callsign -- generic matched by any (known) call-sign; MAGE cannot recognize call-signs without being told

DE -- "this is" or "from"

delim -- generic matching any delimiter: break or punctuation

GR -- "There will be - - - code-groups or English words in next message"

GRPS -- "groups"

K -- "end of transmission"; a pro-sign

location -- generic matched by any (known) location

MSG -- "message"

new-speaker -- denotes speaker change

---

[18] Section 2.2

NR -- "number"

PSE -- "please"

QRX -- "I will call you again at - - - hours" or, if followed "?", "When will you call me again?"; although MAGE knows the spelling and topical associations of sixty q-signs, the q-signs listed here are the only ones that MAGE knows how to use in context (because they appear as transition symbols in the core grammar)

QRZ -- "You are being called by - - - (on frequency - - -)", or "Who is calling me?"; parentheses indicate an optional argument

QSA -- "The strength of your signals is - - - ", or "What is the strength of my signals?"

QSL -- "I am acknowledging receipt (of - - -)", or "Can you acknowledge receipt (of - - -)?"

QTC -- "I have - - - messages for you", or "How many messages have you to send?"

RPT -- "repeat"

SK -- "end of contact"; a pro-sign

ZOH -- "There will be - - - code-groups in the next message"

Subnetworks

Legend:

- States are represented by circles and transitions by arrows.
- A circle containing an *S* represents the subnetwork's start-state. Any circle with a darkened area represents a terminal state.
- Each transition has one or more transition symbols. If a transition has more than one symbol, they are separated by commas.
- A word composed of upper-case letters surrounded by (single)

quotation marks indicates that this transition accepts the particular chatter word.

- A word composed of upper-case letters, but not surrounded by quotation marks, denotes a push (call) to the named subnetwork.
- "(new-speaker)" denotes a speaker change, or switch of receiving and sending operators
- Other words composed of lower-case letters, and " # ", denote generic tokens that are replaced by specific chatter words at parse-time (e.g., "callsign" may be replaced by "ROCK", an operator's call-sign).

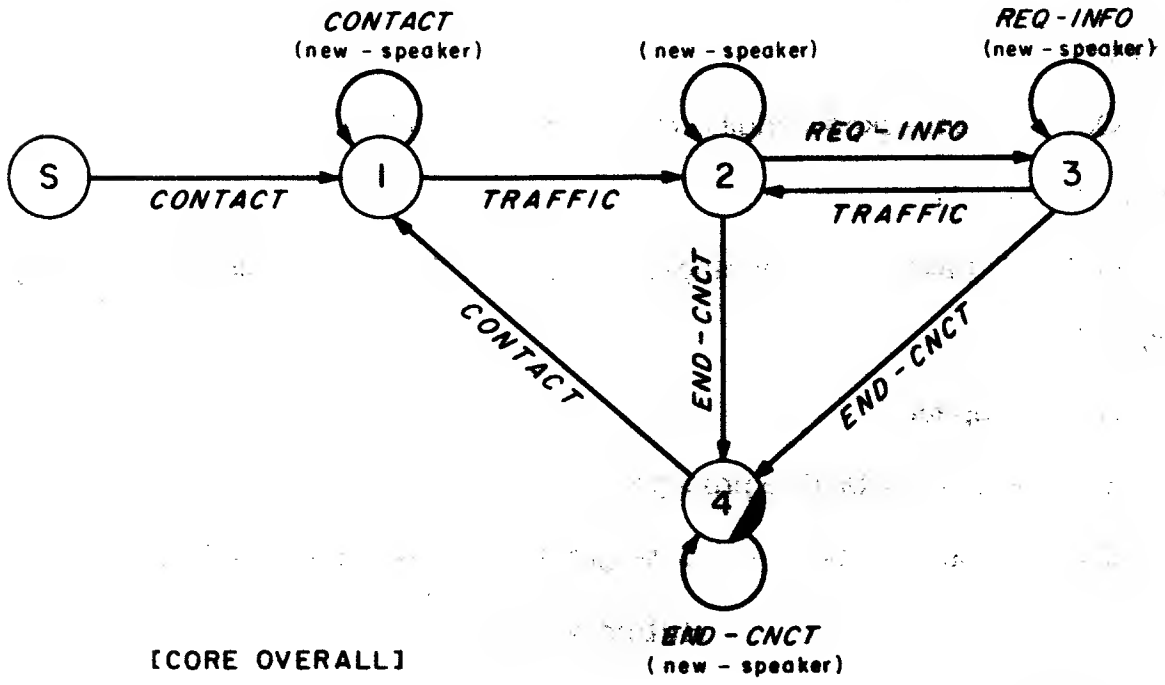


Figure 17: **OVERALL subnetwork**

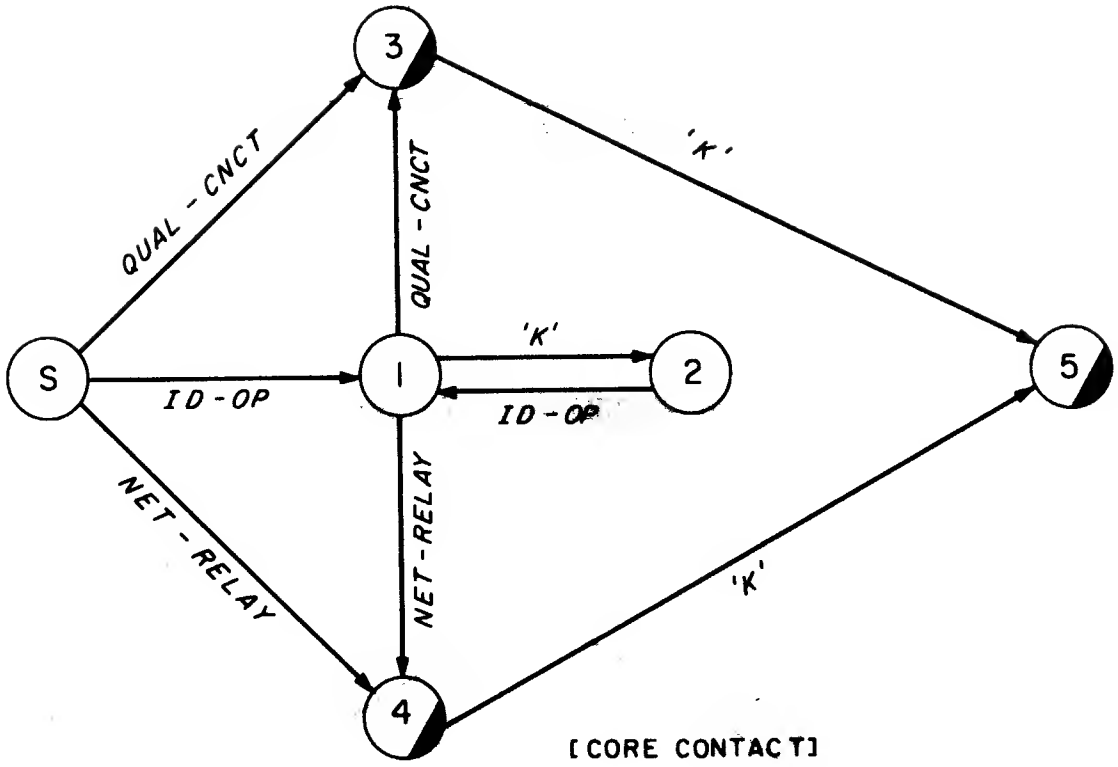
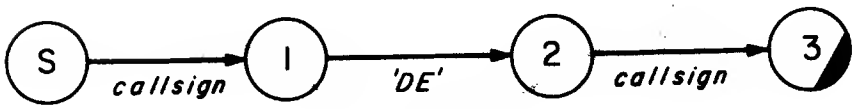
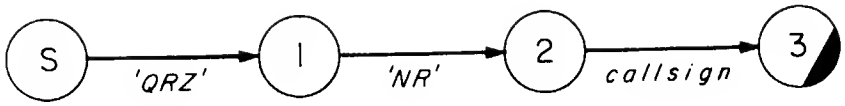


Figure 18: **CONTACT subnetwork**



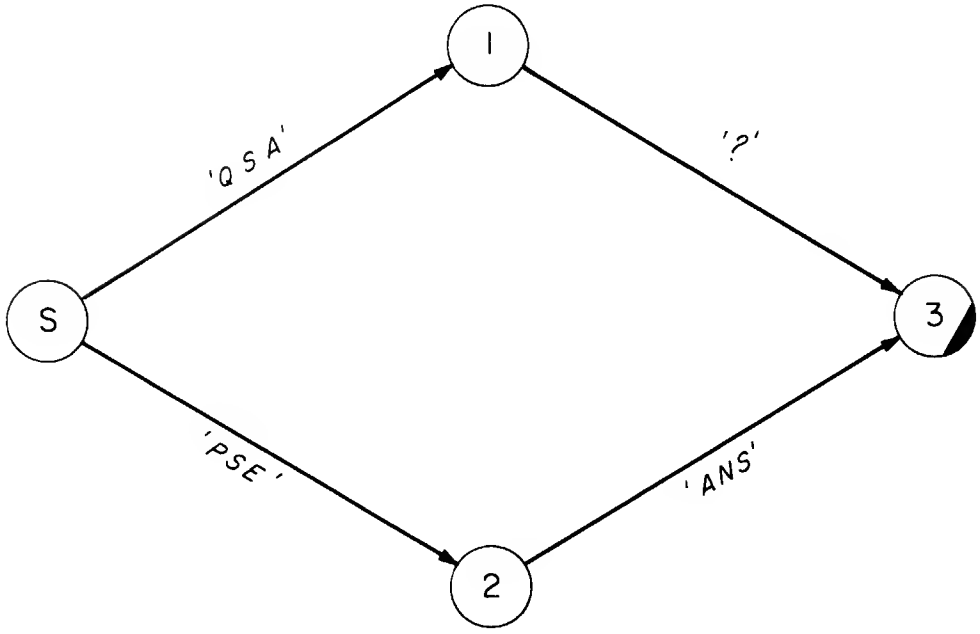
[CORE ID-OP]

Figure 19: **ID-OP subnetwork**



[CORE NET-RELAY]

Figure 20: **NET-RELAY** subnetwork



[CORE QUAL-CNCT]

Figure 21: **QUAL-CNCT** subnetwork

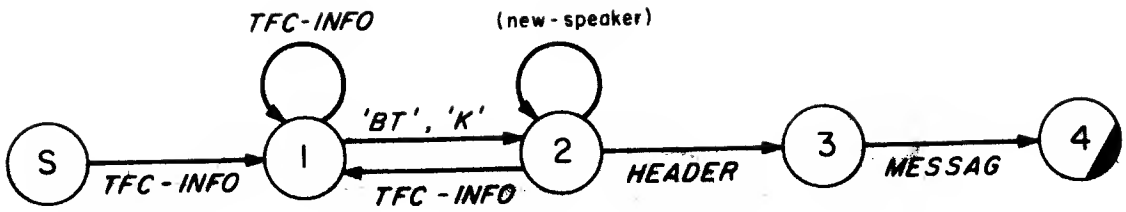


Figure 22: **TRAFFIC subnetwork**

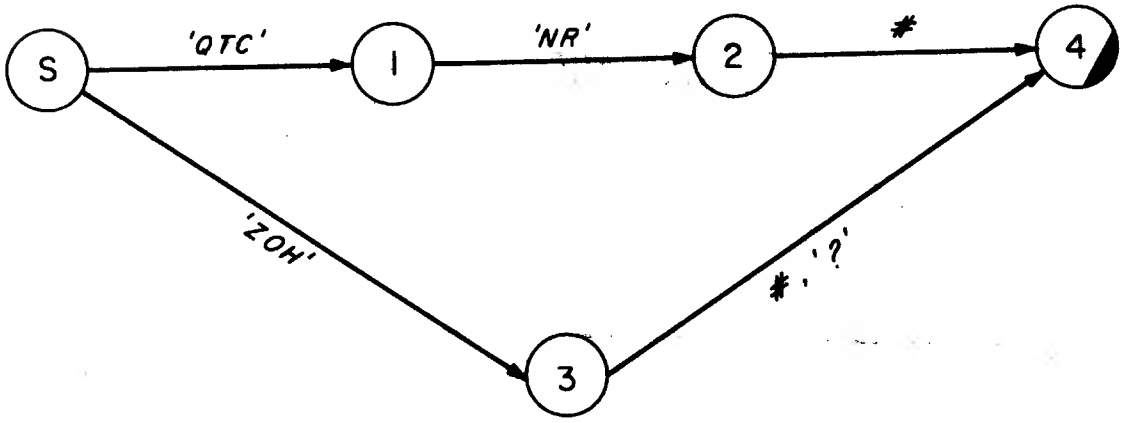


Figure 23: **TFC-INFO subnetwork**

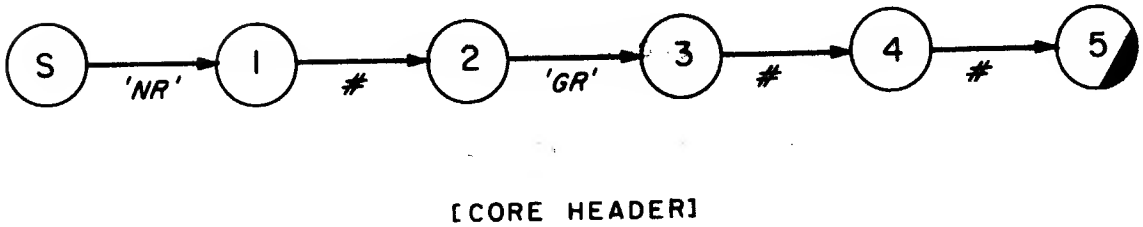
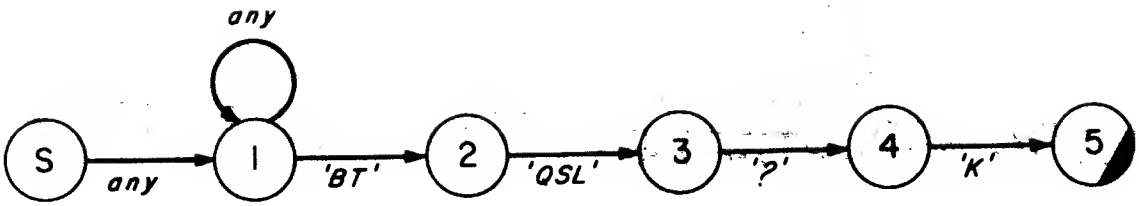
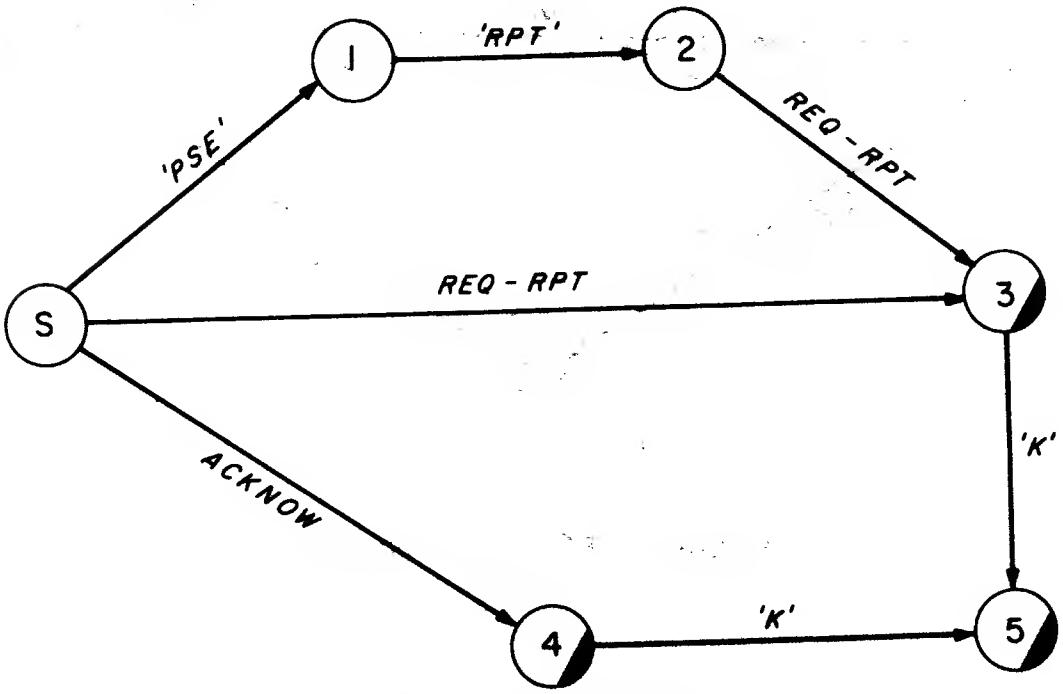


Figure 24: **HEADER subnetwork**



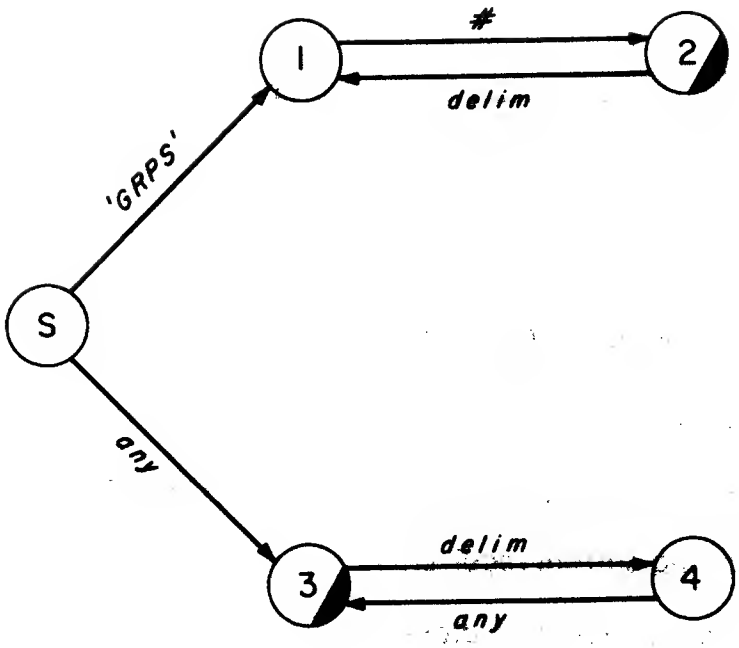
[CORE MESSAG]

Figure 25: **MESSAG subnetwork**



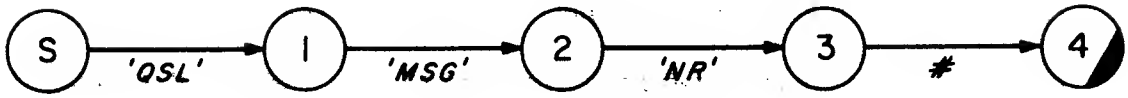
[CORE REQ - INFO]

Figure 26: **REQ-INFO subnetwork**



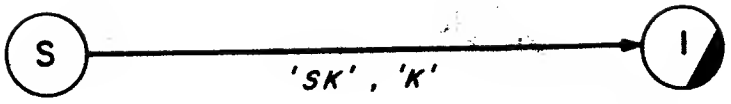
[CORE REQ-RPT]

Figure 27: **REQ-RPT subnetwork**



[CORE ACKNOW]

Figure 28: **ACKNOW subnetwork**



[CORE END - CNCT]

Figure 29: **END-CNCT subnetwork**

Registers

<information-about-receiving-operator> -- Call-sign, location of station, and other information regarding current receiver.

<information-about-sending-operator>

<last-word-received> -- Useful for error-recovery.

<time-and-date>

<scratch-pad> -- Temporary storage for saving arguments, etc.

<number-of-words-in-message>

<id-number-of-message> -- Usually numbered in order of sending.

<number-of-words-received-so-far-in-message> -- Useful for comparing with contents of *<number-of-words-in-message>* register to determine whether entire message has been received.

<last-word-received-in-message> -- Useful for error-recovery.

<general-situation-description> -- Description of radio-network status.

<quality-of-contact> -- Description of station status. There is one of these registers for each active operator.

<expected-action> -- Actions that an operator is expected to perform, usually

in response to request; this provides a context for unpredictable actions. There is one of these registers for each operator.

<pending-questions> -- Questions an operator is expected to answer; this provides a context for unpredictable phrases that might be answers to questions. There is one of these registers for each active operator.

<requests-for-repeats> -- Requests for something (usually a code-group) to be repeated.

Tests

[GROUP? input] -- Returns TRUE if the argument is not a q-sign or delimiter; used only when transition symbol is "any".

[NOT? *<list>*] -- Returns TRUE if the input word is not a member of *<list>*; used when transition symbol is "any". The argument 'input' does not appear explicitly in this test specification because test and action specifications are constrained to include only one argument; however, the actual functions that implement these tests and actions also have access to the set of context registers and the current input token.

[~RECEIVER? input] -- Returns TRUE if token is not (due to '~') the same as the call-sign in the *<information-about-receiving-operator>* register; used only when transition symbol is "callsign".

Actions

[RECEIVER input] and [SENDER input] -- Put input token in call-sign field of *<information-about-receiving-operator>* or *<information-about-sending-operator>* register, respectively; symbol is "callsign".

[NSPEAK T] -- Switch contents of *<information-about-receiving-operator>* and

<information-about-sending-operator> registers, if non-empty; symbol is "new-speaker", denoting speaker change.

[SCRATCH input] -- Put input token in the *<scratch-pad>* register, destroying previous contents; symbol arbitrary.

[Q-VAL input] -- Add input token to list of tokens in *<scratch-pad>* register without destroying previous contents: the first element of list is the pivot word, others are its arguments; symbol arbitrary.

[Q-ACT SCRATCH] -- Get pivot word (usually q-sign) and arguments from *<scratch-pad>* register and put in one of the *<quality-of-contact>*, *<expected-actions>*, or *<general-situation-description>* registers, depending on meaning of pivot word and its argument(s); symbol arbitrary but always preceded directly or indirectly by a pivot word.

[Q-ACT input] -- The particular pivot word is not likely to have arguments, so proceed to put it in one of the above registers; symbol usually a q-sign.

[Q-PEND SCRATCH] -- Get pivot word from the *<scratch-pad>* register and put in the *<pending-question>* or *<expected-action>* register, depending on the meaning of pivot word; symbol is "?".

[MSG-NUM input] -- Put token in *<id-number-of-message>* register; this is the identification number of the next message; symbol is " # ".

[TFC-GR-NUM input] -- Put token in *<number-of-words-in-message>* register; this is the number of code-groups or English words to be sent in the next message; symbol " # ".

[GMT-TIME input] -- Put token in time field of *<time-and-date>* register; this is time of transmission of most recent message; symbol " # ".

[ADD-GROUP input] -- Put token in the *<last-word-received-in-message>* register, useful for error-recovery, and increment the *<number-of-words-received-so-far-in-message>* register; symbol "any".

[LAST-GROUP T] -- Compare contents of the *<number-of-words-received-so-far-in-message>* with contents of *<number-of-words-in-message>* register; if former $\leq$ latter, tell COMDEC to turn off its code-group recognition mechanism; symbol is "BT" or some other break.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MIT/LCS/TR-233	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Automatic Extension of an Augmented Transition Network Grammar for Morse Code Conversations		5. TYPE OF REPORT & PERIOD COVERED B.S.Thesis/May 1979 6. PERFORMING ORG. REPORT NUMBER MIT/LCS/TR-233
7. AUTHOR(s) Gail E. Kaiser		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0661
9. PERFORMING ORGANIZATION NAME AND ADDRESS MIT/Laboratory for Computer Science 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS ARPA/Department of Defense 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE April 1980 13. NUMBER OF PAGES 99
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) ONR/Department of the Navy Information Systems Program Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

[redacted]d transition networks
[redacted] acquisition
[redacted] code

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a [redacted] that [redacted] much of the knowledge required by a parsing system that processes conversations in a 'natural' language akin to ham-radio jargon. The learning program derives information from [redacted] and uses this knowledge to extend the 'core' augmented transition network (ATN) grammar. The

DD [redacted] EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS [redacted] Data Entered)

parser can use the extended grammar to process the example sentences, plus a large number of syntactically and semantically related sentences.

The learning program uses a set of heuristics to determine the difference between the existing version of the grammar and a superset that could process the example sentence. A set of models act as templates to produce possible extensions to the grammar. An evaluation measure selects one of the extensions and adds it to the grammar. This extension is henceforth an integral component of the knowledge base and may be used by the parser to process conversations and by the learning program to extend the grammar further.

This report relates the mechanisms used by the learning program to grammatical inference of context-sensitive languages, which include the natural languages, and some proposed linguistic models of human language acquisition. These models describe language acquisition as a process of developing hypotheses according to the constraints of innate universal rules, and acceptance of those hypotheses that make it possible for the child to understand new sentences. Similarly, the learning program develops its hypotheses within the constraints of certain 'universal' models and accepts only those hypotheses that enable the parser to process the motivating example.